Topological Data Analysis of Real Algebraic Varieties

Candidate number: 1000778

University of Oxford

A dissertation submitted for the degree of

Master of Science

Summer 2016

Acknowledgements

I would like to thank my advisers Heather Harrington and Emilie Dufresne for their guidance and support, without which this project would not have been possible. I would also like to thank Jon Hauenstein for helpful suggestions regarding `Bertini`, Nina Otter for providing useful pointers on computing persistent homology, and Bernd Sturmfels for the quartics used in Section 4.2.

Abstract

In this dissertation we present new methods for studying real algebraic varieties using topological data analysis. Topological data analysis (TDA) provides a growing body of tools for computing geometric and topological information about spaces from a finite sampling of points. Real algebraic varieties are subspaces of Euclidean space which consist of points which evaluate to zero on a system of polynomials.

We start by describing a new adaptive algorithm for finding provably dense samples of points on real algebraic varieties given a set of defining polynomials. The algorithm utilizes methods from numerical algebraic geometry to give formal guarantees about the density of the sampling. A tailored variant of the algorithm is discussed that employs heuristic methods to minimize the number of points in the sample while still providing provably dense output. This sample minimization makes applying TDA methods feasible, since TDA methods consume significant computational resources that scale poorly in the number of sample points. We also describe how to extend the algorithm to real semialgebraic sets, which can contain polynomial inequality constraints as well as equalities.

Using an implementation of the algorithm, we then apply TDA to real several algebraic varieties. Results indicate that the specially tailored version of the algorithm significantly reduces the number of points in a sample. For some examples, TDA and the provable density of the algorithm's output combine to give theoretical justification for the existence of topological features. In others, TDA yields geometric information about the variety. We then detail a strategy for applying the algorithm and TDA to determine information about the configuration space of the molecule cylcooctane (C_8H_{16}). Source code for the sampling algorithm and examples is made publicly available.

Contents

List of Figures

Chapter 1

Introduction

Topological data analysis (TDA) comprises a rapidly growing body of theoretical framework, specialized algorithms, and software packages for studying the shape and topological features of data sets. The theory gives strong guarantees: if a dense set of points from a topological space is provided as input, TDA algorithms capture most of the topological information about the space. TDA has been applied successfully to fields as diverse as signal processing, natural images, neurology, and sensor networks [24, 7, 12]. Though TDA methods provide powerful insight when applied properly, they require substantial computational resources to execute.

Algebraic varieties are the zero sets of polynomials system in several variables. Varieties, particularly those consisting of *real* solutions to polynomial equations, arise in applied settings like chemistry and kinematics. Constrained motion of atoms in molecules for the former, and of robotic platforms for the latter, produce models with polynomial constraints. Obtaining topological insights about varieties can translate to information about a robot's range of motion, or how a molecule can bend and change shape.

Leveraging TDA to study real algebraic varieties requires a strategy for computing real solutions to the variety's defining polynomial equations. In addition, the points fed into TDA algorithms need to be densely sampled from the variety to form strong theoretical conclusions. TDA's computational resource consumption scales exponentially with the number of input points, however, which encourages keeping that number low.

This dissertation introduces and implements a sampling algorithm for real algebraic varieties that balances these two objectives. To match the strong theoretical guarantees of the TDA framework, the algorithm outputs provably dense samplings of points from a variety given the variety's defining polynomials. In some cases, this allows results from TDA to serve as full theoretical proof of the existence of a variety's topological features. To meet the objective of keeping computational costs for TDA tractable, the algorithm employs heuristics to judge when a point should be added to the sample. The implementation of the sampling algorithm ties together multiple software packages tailored for numerical algebraic geometry, topological data analysis, and indexing data by its location in space.

The rest of this dissertation is organized as follows. Chapter 2 gives an introduction to the theory of persistent homology, topological data analysis, algebraic geometry, and numerical algebraic geometry (NAG). Chapter 3 describes the sampling algorithm and implementation details. Chapter 4 presents the output of TDA computations performed with results from the sampling algorithm as input. Chapter 4 also discusses a potential application of the algorithm to the problem of analyzing the configuration space of cyclooctane, an organic molecule.

Chapter 2

Background

2.1 Numerical algebraic geometry

2.1.1 Algebraic geometry

The central objects of study in classical algebraic geometry are the zero sets of systems of polynomials. In principle, the polynomials can be defined over any field, though applications typically concentrate on the familiar fields $\mathbb{R}$ and $\mathbb{C}$. The following definition fixes some notation and terminology going forward.

Definition 2.1. Let k be a field. If S is a subset of $k[x_1, \ldots, x_N]$, then the *(affine) algebraic variety* corresponding to S is the set of $x \in k^N$ such that $f(x) = 0$ for all $f \in S$, denoted $\mathcal{V}(S)$. If $S = \{f_1, \ldots, f_n\}$ is finite we also denote this set $\mathcal{V}(f_1, \ldots, f_n)$. In the case that $k = \mathbb{C}$, the *real algebraic variety* corresponding to S is $\mathcal{V}(S) \cap \mathbb{R}$, denoted $\mathcal{V}_{\mathbb{R}}(S)$. In the case that $k = \mathbb{C}$ or $\mathbb{R}$, an isolated solution of a variety $V \subseteq k^N$ is a point in V which is isolated in the Euclidean topology on V.

An algebraic variety is called *irreducible* if it is not the union of two proper sub-algebraic varieties. Given an algebraic variety V, the set $\mathcal{I}(V) \subseteq k[x_1, \ldots, x_N]$ is the set of polynomials $f \in k[x_1, \ldots, x_N]$ where $f(x) = 0$ for all $x \in V$.

There is a strong connection between varieties defined by functions in a polynomial ring and ideals in the ring. Recall that for $S \subseteq k[x_1, \ldots, x_N]$, the set $\langle S \rangle$ consists of polynomials of the form $f = \sum_{i=1}^{n} h_i s_i$ where each s_i is an element of S and each h_i is in $k[x_1, \ldots, x_N]$. It is straightforward to check that $\langle S \rangle$ is an ideal

and $S \subseteq \langle S \rangle$. If an ideal I is equal to $\langle S \rangle$ for some subset $S \subseteq I$, then S is called a *basis* for I. The whole polynomial ideal $\langle S \rangle$ vanishes on $\mathcal{V}(S)$.

Proposition 2.2. *Let $S \subseteq k^N$ for some field k. Then $\mathcal{V}(S) = \mathcal{V}(\langle S \rangle)$.*

Proof. Suppose $x \in \mathcal{V}(\langle S \rangle)$ and $f \in S$. Clearly $f \in \langle S \rangle$, so $f(x) = 0$. Thus $\mathcal{V}(\langle S \rangle) \subseteq \mathcal{V}(S)$. Suppose $x \in \mathcal{V}(S)$ and $f = \sum_{i=1}^{n} h_i s_i \in \langle S \rangle$. Then $f(x) = \sum_{i=1}^{n} h_i(x) s_i(x) = \sum_{i=1}^{n} h_i(x) \cdot 0 = 0$, so $\mathcal{V}(S) \subseteq \mathcal{V}(\langle S \rangle)$. □

It may appear that replacing S with the potentially larger set $\langle S \rangle$ actually complicates rather than simplifies the description of a variety. Hilbert's Basis Theorem shows, however, that every polynomial ideal has a finite basis set. This theorem and Proposition 2.2 imply that any variety is the set of points $x \in k^N$ on which some list of polynomials $f_1, \ldots, f_n$ vanish. Hilbert's Basis Theorem can also be used to prove some other basic facts characterizing varieties.

Proposition 2.3. *Let k be a field, $S, T \subseteq k[x_1, \ldots, x_N]$, $A = \mathcal{V}(S)$, and $B = \mathcal{V}(T)$. Then:*

1. *$\mathcal{I}(A)$ is an ideal*
2. *$\mathcal{V}(\mathcal{I}(A)) = A$*
3. *$A \cup B$ and $A \cap B$ are also algebraic varieties defined by finitely many equations*
4. *A is irreducible if and only if $\mathcal{I}(A)$ is prime*

Proof. By Hilbert's Basis Theorem and Proposition 2.2 we can assume without loss of generality that S and T are finite.

1. Let $f, g \in \mathcal{I}(A)$ and $h \in k[x_1, \ldots, x_N]$. Then for $x \in A$, clearly $(f + g)(x) = f(x) + g(x) = 0$, so $f + g \in \mathcal{I}(A)$. Also $fh(x) = 0 \cdot h(x) = 0$, so $fh \in \mathcal{I}(A)$.

2. Let $x \in \mathcal{V}(\mathcal{I}(A))$. Then since $S \subseteq I(A)$ follows by definition, it follows that $f(x) = 0$. Thus $x \in \mathcal{V}(S) = A$. Similarly, if $x \in A$, then $f(x) = 0$ for every $f \in \mathcal{I}(A)$ by definition, so $x \in \mathcal{V}(\mathcal{I}(A))$. Thus $\mathcal{V}(\mathcal{I}(A)) = A$.

3. For the first equality, note that $x \in A \cap B$ if and only if $f(x) = 0 = g(x)$ for any polynomials f and g in A and B respectively if and only if $x \in \mathcal{V}(S \cup T)$. We claim $A \cup B = \mathcal{V}(\{fg \mid f \in S, g \in T\})$. If $x \in A \cup B$ and both $f \in S$ and $g \in T$, then either $f(x) = 0$ or $g(x) = 0$, so $fg(x) = 0$. If $x \in \mathcal{V}(\{fg \mid f \in S, g \in T\})$ then suppose, to the contrary, $x \notin A \cup B$. Then there are $f \in S$ and $g \in T$ such that $f(x), g(x) \neq 0$. Since k is a field, this implies $fg(x) \neq 0$, a contradiction.

4. ($\Rightarrow$) We prove the contrapositive. Suppose $fg \in \mathcal{I}(A)$ and both $f, g \notin \mathcal{I}(A)$. For any $x \in A$, either $f(x) = 0$ or $g(x) = 0$ by definition. Since both f and g are not in $\mathcal{I}(A)$, this implies there are $x_1, x_2 \in A$ where $f(x_1) = g(x_2) = 0$ and $f(x_2), g(x_1) \neq 0$ That is to say, $f^{-1}(0) \not\subseteq g^{-1}(0)$ and $g^{-1}(0) \not\subseteq f^{-1}(0)$. Thus $A = (\mathcal{V}(f) \cap A) \cup (\mathcal{V}(g) \cap A)$ presents A as a union of proper algebraic varieties by item 3.

   ($\Leftarrow$) Suppose $\mathcal{I}(A)$ is prime and, to the contrary, $A = C \cup D$ for some proper algebraic subvarieties of A. Note that $I(C) \subseteq I(D)$ would imply $C \subseteq \mathcal{V}(I(C)) \subseteq \mathcal{V}(I(D)) = D$ by item 3. This contradicts that A is the union of two proper subsets, and so $I(C) \not\subseteq I(D)$. A symmetric argument gives $I(D) \not\subseteq I(C)$. It follows, then, that some $f \in I(C) - I(D)$ exists, and similarly some $g \in I(D) - I(C)$ exists. Equivalently, both f and g are not in $\mathcal{I}(A) = I(C \cup D)$, but fg is, which shows $\mathcal{I}(A)$ is not prime. This is a contradiction, so A is irreducible.

□

The previous propositions establish a close connection between varieties and ideals, but two different ideals can describe the same variety. The ideals $\langle x \rangle$ and $\langle x^2 \rangle$ serve as a good example of this, since $\mathcal{V}(\langle x \rangle) = \{0\} = \mathcal{V}(\langle x^2 \rangle)$ in $k[x]$ if k is of characteristic 0. Hilbert's Nullstellensatz broadens this connection into a characterization. Notice that for an algebraic variety V, a polynomial f is in $\mathcal{I}(V)$ if $f^k \in \mathcal{I}(V)$, since $(f(x))^k = 0$ implies $f(x) = 0$. Ideals with this property are important enough to name.

Definition 2.4. Given an ideal I in a polynomial ring $k[x_1, \ldots, x_N]$ over the field k, I is a *radical ideal* (or just *radical*) if $f^k \in I$ implies that $f \in I$. The set of functions g such that $g^k \in I$ for some positive k is denoted $\sqrt{I}$.

It can be checked that $\sqrt{I}$ is a radical ideal for any I (Lemma 4.5, [10]). The previous discussion makes clear that for any variety $V = \mathcal{V}(I)$ we have $\mathcal{I}(V) \subseteq \sqrt{I}$. Hilbert's Nullstellensatz states that the other inclusion, $\sqrt{I} \subseteq \mathcal{I}(V)$, is also true if the field k is algebraically closed. Combining this information with item 2 of Proposition 2.3 gives an exact correspondence between algebraic objects (radical ideals), and geometric objects (algebraic varieties).

Theorem 2.5 (Hilbert's Nullstellensatz)**.** *Let I be an ideal in the polynomial ring $k[x_1, \ldots, x_N]$ where k is an algebraically closed field. Then $\mathcal{I}(\mathcal{V}(I)) = \sqrt{I}$.*

This correspondence suggests that symbolic methods for manipulating polynomial ideals can be adapted to yield geometric information about complex algebraic varieties. For real algebraic varieties, however, the situation is not as clean since $\mathbb{R}$ is not algebraically closed, and the Nullstellensatz does not apply. Numerically oriented methods are thus particularly attractive as an approach for analyzing real algebraic varieties.

The union and intersection properties from item 3 of Proposition 2.3 allow the construction of a topology on k^N, with (closed) basis elements consisting of irreducible varieties. This topology is called the Zariski topology on k^N. In the cases where $k = \mathbb{C}$ or $k = \mathbb{R}$, for any polynomial $f \in k[x_1, \ldots, x_N]$ the set $\mathcal{V}(f) = f^{-1}(0)$ is closed in the standard topology, so the Zariski topology is coarser than the standard topology. It is strictly coarser, in fact. A closed interval in $\mathbb{R}$ is not Zariski closed, for instance, since by the Fundamental Theorem of Algebra any Zariski closed subset of $\mathbb{R}$ is finite.

Recall that for any topological space X, a property concerning elements of X is said to hold generically on X if there is an open dense subset of X where the property holds for each element in the subset. The likelihood of choosing a point in $\mathbb{C}^N$ or $\mathbb{R}^N$ at random that fails to have a generic property is negligibly small. Most of the properties explored in numerical algebraic geometry hold generically, but not at every point in the space under consideration. The probability based argument

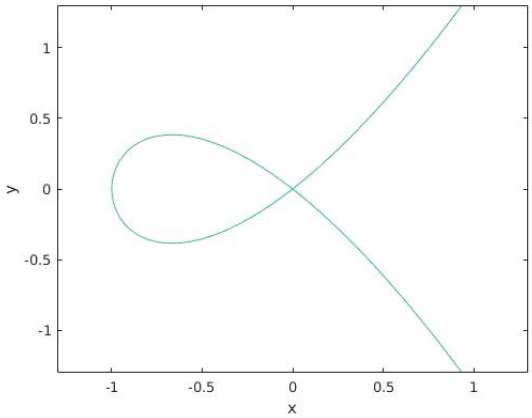


Figure 2.1: The real algebraic variety $\mathcal{V}_{\mathbb{R}}(f)(y^2 - x^2(x+1))$.

is of vital importance to the practical computability of NAG methods which rely on generic properties. Choosing a point at random is computationally inexpensive, whereas formal theoretical verification that a property holds is potentially difficult.

Real and complex algebraic varieties have greater geometric complexity than smooth manifolds in general. Consider the real algebraic variety $\mathcal{V}_{\mathbb{R}}(y^2-x^2(x+1))$ pictured in Figure 2.1. The point on the variety at $(0,0)$ has no open neighborhood which is homeomorphic to $\mathbb{R}$, though all the other points on the variety do. Points like $(0,0)$ in this simple example motivate the following definition. Recall that for a system of polynomial functions $f_1, \ldots, f_n \in k[x_1, \ldots, x_N]$, the *Jacobian* of the system, Jf, is an $n \times N$ matrix with entries in $k[x_1, \ldots, x_N]$:

$$Jf = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_N} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_N} \end{pmatrix}$$

Definition 2.6. Let $V \subseteq \mathbb{C}^N$ be a non-empty irreducible algebraic variety and $f_1, \ldots, f_n \in k^N$ be some basis polynomial system where $V = \mathcal{V}(f_1, \ldots, f_n)$. The *dimension* of V is the length d of the longest chain $V_d \supsetneq V_{d-1} \supsetneq \cdots \supsetneq V_0$ of distinct nonempty irreducible subvarieties of V. If Jf is its Jacobian matrix, a point $x_0 \in V$ is *singular* if the rank of Jf when evaluated at x_0 is less than $N - d$. Otherwise x_0 is *nonsingular*.

It can be shown that whether points are singular or not does not depend on the basis of polynomials chosen for the variety, and that the set of singular points of a variety is a proper subvariety [22]. Geometrically (for $k = \mathbb{C}, \mathbb{R}$), singular points are the points on a variety in $\mathbb{C}^N$ or $\mathbb{R}^N$ that have a tangent space with a dimension larger than the variety's dimension. In the previous example, the point $(0, 0)$ has a 2 dimensional tangent space on the variety where every other point has a 1 dimensional tangent space. Nonsingular points are also called *manifold* points, and a result due to Whitney shows that the nonsingular points of a real or complex irreducible algebraic variety form a smooth manifold of the same (real or complex, as appropriate) dimension as the variety [28].

Apart from regions of self-intersection in irreducible varieties, intersections of proper subvarieties in reducible algebraic varieties also give rise to singular regions. As an example, the variety $\mathcal{V}(xy)$ consists of the lines $x = 0$ and $y = 0$ which intersect at a point $(0, 0)$. The point $(0, 0)$ does not have a clearly defined tangent line, while all other points of the variety do. How many ways can we decompose a reducible algebraic variety into a union of irreducible subvarieties? Perhaps surprisingly, the answer to this question is "only one, up to containment" as recorded in the following theorem.

Theorem 2.7 (Theorem 6.4, [10])**.** *Let $V \subset k^N$ be an algebraic variety. Then V has a decomposition*

$$V = V_1 \cup \cdots \cup V_m$$

such that each V_i is irreducible, $V_i \not\subseteq V_j$ for $i \neq j$, and this decomposition is the unique decomposition with these properties up to reordering of the V_i.

This decomposition is called the *irreducible decomposition* of a variety V, and each V_i is an *irreducible component* of V. It allows us to extend the notions of singular points and dimension to reducible varieties. The *dimension* of V is the largest dimension of one of V's irreducible components. If V has dimension greater than 0 then it is called *positive dimensional*. A point in V is nonsingular if it is a nonsingular point of exactly one irreducible component of V. Otherwise it is singular. If V contains only d-dimensional irreducible components, then V is *pure d-dimensional*.

2.1.2 Homotopy continuation

Methods in numerical algebraic geometry (NAG) look to estimate solutions to systems of polynomial equations via a numerical rather than symbolic approach. One of the most effective such methods is homotopy continuation, a process which efficiently approximates all of the isolated solutions of an algebraic variety given the variety's defining polynomials as input. We give an overview of homotopy continuation here, while the sources [26] and [2] discuss in detail the theory and technical considerations necessary for implementation. In all cases, the solutions obtained are only approximations. The underlying theory guarantees, however, that the approximation methods used will converge to a solution when applied iteratively to an approximate solution, and the approximation methods can be carried to any desired degree of precision.

Given a polynomial system with real coefficients $f_1, \ldots, f_N$, it can be expressed as $f : \mathbb{C}^N \to \mathbb{C}^N$ where $f(x) = (f_1(x), \ldots, f_N(x))^T$. We denote $\mathcal{V}(f_1, \ldots, f_N)$ as $\mathcal{V}(f)$. Let $f, g : \mathbb{C}^N \to \mathbb{C}^N$ be polynomial systems whose constituent polynomials have real coefficients and such that both $\mathcal{V}(f)$ and $\mathcal{V}(g)$ are 0 dimensional. Though we have restricted our consideration to "square" systems with the same number of variables and polynomials, systems with more polynomials than variables can be converted into square systems whose isolated solutions contain all the isolated solutions to the original system (see Section 1.1.4, [1]). Some solutions to this reduced system may not be solutions of the original, but these can be filtered out. We will call f the "target system" whose isolated solutions we are seeking to approximate, and g the "start system" which has at least as many already known isolated solutions as f has isolated solutions. The first step of homotopy continuation considers f and g as embedded into a family of polynomial systems parameterized by $\mathbb{C}$. That is there exists $H : \mathbb{C}^N \times \mathbb{C} \to \mathbb{C}^N$, where $f(z) = H(z, 0)$ and $g(z) = H(z, 1)$. Furthermore, we assume that $H(z, t)$ is complex analytic in t, that $p(z) = H(z, t)$ for any fixed t is a polynomial system, that we know a set S of solutions of g, and that every solution of g in S is trackable.

Definition 2.8. (from [16]) A solution x of g is *trackable* using $H(z, t)$ from $t = 1$ to $t = 0$ if there exists a smooth function $x^* : (0, 1] \to \mathbb{C}^N$ such that $x^*(1) = x$ and $x^*(t)$ is a nonsingular isolated solution of $H(x, t) = 0$ for all $t \in (0, 1]$.

The key idea for homotopy continuation is to track the solution paths numerically for each solution of g in S as t goes from 1 to 0. To do this we must assume that none of the solution paths beginning at different solutions in S cross. That is to say that if $p_1, p_2 : (0,1] \to \mathbb{C}^N$ are two different solution paths, there is no $t^* \in (0,1]$ such that $p_1(t^*) = p_2(t^*)$. As $t \to 0$, a solution path p can either *diverge* if $\|p(t)\| \to \infty$ as $t \to 0$, or otherwise *converge* to a solution of f as $t \to 0$. Such a solution might be isolated or not, in principle. Finding f's isolated solutions requires defining start systems and homotopies which fit all of the technical requirements on having good solution paths, and also where the solutions of f at the end of the solution paths contain all the isolated solutions of f. One of the simplest appropriate choices for H is a "total degree homotopy".

Theorem 2.9 (Theorem 8.4.1 [26])**.** *Let $f = (f_1, \ldots, f_N)^T$ be a polynomial system, d_i be the degree of f_i, and $g(z)$ be the polynomial system $g = (z_1^{d_1} - 1, \ldots, z_N^{d_N} - 1)^T$ which has $d = \Pi_{i=1}^N d_i$ nonsingular solutions. Then the d solution paths of the homotopy*

$$H(z,t) := \gamma t g(z) + (1-t) f(z)$$

starting at the solutions of $g(z) = 0$ are nonsingular for $t \in (0,1]$ and the endpoints of the solution paths defined by the homotopy include all of the nonsingular solutions of $f(z)$ for generic $\gamma \in \mathbb{C}$. In particular, there are a finite number of exceptions when γ is restricted to the unit circle $\gamma = e^{i\theta}$.

What remains is to give a numerical method for calculating the endpoints of these solution paths. The most computationally feasible method is called "path tracking". If $H_z(z,t)$ and $H_t(z,t)$ are the partial derivatives of H with respect to z and t, the following differential equation due to Davidenko [11] holds for a solution path $x^*(t)$:

$$H_z(x^*(t),t)\frac{dx^*(t)}{dt} + H_t(x^*(t),t) = \frac{dH(x^*(t),t)}{dt} = 0$$

Since $x^*(t)$ is nonsingular for all t in $(0,1]$, the matrix $H_z(x^*(t),t)$ is invertible. The above equation can thus be rearranged into $\frac{dx^*(t)}{dt} = -(H_z(x^*(t),t))^{-1} H_t(x^*(t),t)$. Applying Euler's method gives $x^*(t+\Delta t) \approx x^*(t) + \Delta t \frac{dx^*(t)}{dt}$. This yields an estimate x that can be sharpened using Newton's method. One step of Newton's

method replaces the current estimate x with $x-[H_z(x,t+\Delta t)]^{-1}H(x,t)$. Repeated iterations of Newton's method usually result in a much more accurate estimate of $x^*(t+\Delta t)$. These two methods suggest a simplified path tracking "estimator-predictor" procedure for each solution path x^* starting at $t=1$:

1. Select some step size Δt which produces sufficiently small estimated error when applying Euler's method to the solution path
2. Estimate $x^*(t+\Delta t)$ using Euler's method and set $t=t+\Delta t$
3. Apply Newton's method some number of times to improve the estimate for $x^*(t)$
4. Repeat 1-3 until t is very small and the estimated error for $x^*(t)$ is also small. The estimate $x^*(t)$ is an approximation for an isolated solution of f

There are quite a few places where things might go wrong in this procedure. When there are multiple solution paths, overly large step sizes Δt and insufficient sharpening of estimates using Newton's method could result in following the wrong path. If f has a multiple root, then Newton's method will not work very well around it as $t \to 0$. These issues can be resolved, albeit with significant care and some modification of the procedure, as detailed in the references mentioned previously. The core idea remains the same, however.

Total degree homotopies only make use of one parameter variable, but path tracking can be extended to homotopies which have an arbitrary number of parameters. The path in parameter space from $t=1$ to 0 for total degree homotopies becomes a continuous path $\phi : [0,1] \to \mathbb{C}^q$, where q is the number of parameters. If $H' : \mathbb{C}^N \times \mathbb{C}^q \to \mathbb{C}^n$ represents the parameterized family of functions constructed, the homotopy $H(z,t)$ is then defined as $H(z,t) = H'(z,\phi(t))$. If a particular application does not demand full multiplicity information about the isolated solutions of the target system f, a total degree homotopy wastes computational time tracking paths which potentially lead to the same isolated solution of f. To minimize this overhead we can use a "parameter homotopy" procedure.

Theorem 2.10. *(Theorem 7.1.1 [26]) Let $H'(z,p)$ be a system of polynomials in n variables and q parameters. That is $H'(z,p) = (H'_1(z,p), \ldots, H'_n(z,p))^T$ and each H'_i is polynomial in both z and p. If $N(p)$ is the number of nonsingular solutions as a function of p, then:*

1. *$N(p)$ is finite and is the same, say N, for generic $p \in \mathbb{C}^q$*
2. *For all $p \in \mathbb{C}^q$, $N(p) \leq N$*
3. *The property $N(p) = N$ is generic with respect to $p \in \mathbb{C}^q$, say with Zariski closed set P^* where $N(p^*) < N$ for $p^* \in P^*$*
4. *The homotopy $H(z,t) = H'(z,\phi(t))$ with $\phi(t) : [0,1] \to \mathbb{C}^q - P^*$ has N continuous, nonsingular solution paths*
5. *As $t \to 0$, the limits of the solution paths of the homotopy H as $t \to 0$ include all the nonsingular roots of $H(z,0) = 0$*

Using the results of this theorem, the parameter homotopy procedure proceeds in three steps for the parameterized polynomial system $g : \mathbb{C}^N \times \mathbb{C}^q \to \mathbb{C}^N$ with target system $f(z,p_0)$ a member of the parameterized family.

1. Pick a random parameter value $p_1 \in \mathbb{C}^q$ and find the isolated solutions of $f(z,p_1) = 0$ (a total degree homotopy could be used for instance). With probability 1, the number of solutions is equal to the maximum number of solutions of $f(z,p)$ for $p \in \mathbb{C}^q$.
2. Let $\phi : [0,1] \to \mathbb{C}^q$ be the straight line path in $\mathbb{C}^q$ starting at $\phi(1) = p_1$ and ending at $\phi(0) = p_0$. Again, there is 0 probability that ϕ crosses an exceptional parameter value in the range $(0,1]$.
3. Use homotopy continuation to find the nonsingular isolated solutions of $f(z,p_0)$ using the homotopy $H(z,t) = g(z,\phi(t))$.

This procedure has the the major computational advantage of potentially tracking fewer paths than a total degree homotopy. Step 1 also only needs to be performed once for finding solutions for multiple polynomial systems in the parameterized family.

2.1.3 Minimum distance problem

It is not immediately clear whether we can use homotopy continuation to find points on a real algebraic variety of any dimension greater than 0. Recall that for a polynomial system $f : \mathbb{C}^N \to \mathbb{C}^n$, $\mathcal{V}_{\mathbb{R}}(f) = \mathcal{V}(f) \cap \mathbb{R}^N$. Given f and some fixed $y \in \mathbb{R}^N$, consider the problem of finding a point $x \in \mathcal{V}_{\mathbb{R}}(f)$ which minimizes the (squared) distance $\|y-x\|^2$. If we solve this problem by obtaining a solution point $x \in \mathcal{V}_{\mathbb{R}}(f)$, this solution is both a point in the real variety $\mathcal{V}_{\mathbb{R}}(f)$, and the solution also gives information about what region of space around y does not contain any points of $\mathcal{V}_{\mathbb{R}}(f)$. Namely, the open ball of radius $\|x - y\|$ centered at y does not contain points of $\mathcal{V}_{\mathbb{R}}(f)$. Formulated as an optimization program the minimum distance problem becomes:

$$\begin{aligned} &\min_{x \in \mathbb{R}^N} \|y - x\|^2 \\ &\quad \text{subject to } f_1(x) = \cdots = f_n(x) = 0 \end{aligned}$$

One way to approach optimization problems of this type is to find all solutions to the Karush-Kuhn-Tucker (KKT) conditions [19, 20] and select the minimizer from this list of solutions. A closely related set of conditions are the Fritz John conditions (FJ conditions).

Theorem 2.11. *(Fritz John [18]) For any optimization problem of the form*

$$\begin{aligned} &\min_{x \in \mathbb{R}^N} g(x) \\ &\textit{subject to } f_1(x) = \cdots = f_m(x) = 0 \\ &\qquad\qquad h_{m+1}(x) \geq 0, \ldots, h_n(x) \geq 0 \end{aligned}$$

each solution x of the problem satisfies

$$-\lambda_0 \nabla g(x) + \sum_{i=1}^{m} \lambda_i \nabla f_i(x) + \sum_{i=m+1}^{n} h_i(x) = 0$$

for some non-zero vector of real numbers $(\lambda_0, \ldots, \lambda_n)$.

To simplify matters somewhat, let us assume that $V \subseteq \mathcal{V}_{\mathbb{C}}(f)$ is a pure d-dimensional variety. As a technical requirement for some of the following results, we also need that f is a system from $\mathbb{C}^N$ to $\mathbb{C}^{N-d}$. Similarly to the previous section, a system arising in an application may not initially have the correct number of equations. For a system $f : \mathbb{C}^N \to \mathbb{C}^n$ with $n > N$, this can be fixed by replacing the system $f_1, \ldots, f_n$ with a system with $N - d$ polynomials in N variables where each polynomial is a linear combination of the form $\lambda_1 f_1 + \cdots + \lambda_n f_n$ for some randomly chosen complex numbers $\lambda_1, \ldots, \lambda_n$. On a nonempty Zariski open subset of the set of $(N-d) \times n$ matrices (each matrix corresponds to a choice of complex coefficients in the $N-d$ linear combinations), this new system's variety will contain the irreducible components of $\mathcal{V}(f)$ as irreducible components, though it might contain other components as well (Theorem 13.5.1 [26]). Results from these extra components can be filtered out. We assume going forward that f is a system in N variables and $N - d$ equations.

Since all of the equality constraints in our problem and all of their gradients are polynomials, the function $g : \mathbb{C}^{2N-d+1} \to \mathbb{C}^{2N-d+1}$ defined by

$$g(x, \lambda_0, \ldots, \lambda_{N-d}) = \begin{pmatrix} f(x) \\ \lambda_0(x-y) + \sum_{i=1}^{N-d} \lambda_i \nabla f_i(x) \\ \sum_{i=0}^{N} \alpha_i \lambda_i - 1 \end{pmatrix},$$

for some non-zero real α_i, is a polynomial system. The final equation is a technical condition which prevents all the λ_i from being 0. Elements $x \in \mathcal{V}(f)$ with solutions $(x, \lambda) \in \mathcal{V}(g)$ are called *critical points* for the minimum distance problem at y. One additional fact is that for a critical point x, the vector $y - x$ is normal to the tangent surface of $\mathcal{V}(f)$ at x. The system g could still define a positive dimensional variety, particularly if the subvariety of $\mathcal{V}(f)$ consisting of singular points is positive dimensional. The following theorem relies on the theory of infinitesimal perturbations to fix this problem.

Theorem 2.12. *(Hauenstein [16]) Let $\pi : \mathbb{C}^{2N-d+1} \to \mathbb{C}^N$ be the projection function onto the first N coordinates. For a fixed $y \in \mathbb{R}^N - \mathcal{V}_{\mathbb{R}}(f)$, the solution set S to the system g above obtained from a parameter homotopy on the parameterized family (with $z \in \mathbb{R}^{N-d}$ and $\gamma \in \mathbb{C}$):*

$$H(x, \lambda_0, \ldots, \lambda_{N-d}, t) = \begin{pmatrix} f(x) - t\gamma z \\ \lambda_0(x-y) + \sum_{i=1}^{N-d} \lambda_i \nabla f_i(x) \\ \sum_{i=0}^{N-d} \alpha_i \lambda_i - 1 \end{pmatrix}$$

contains in $\pi(S)$ an estimated solution to the minimum distance problem on V for generic z, γ, and α_i.

Obtaining points in V by solving the distance problem in this way is computationally inefficient, since some if not most of the computed critical points will be complex. A simpler homotopy allows the computation of only real critical points, at the loss of being able to solve the optimization problem.

Theorem 2.13. *(Griffin, Hauenstein [14]) Let y be a point in $\mathbb{R}^N$, and let $\alpha_0, \ldots, \alpha_{N-d} \in \mathbb{R}$ be non-zero. If the homotopy $H : \mathbb{C}^N \times \mathbb{C}^{N-d+1} \times \mathbb{C} \to \mathbb{C}^{2N-d+1}$ given by*

$$H(x, \lambda_0, \ldots, \lambda_{N-d}, t) = \begin{pmatrix} f(x) - tf(y) \\ \lambda_0(x-y) + \sum_{i=1}^{N-d} \lambda_i \nabla f_i(x) \\ \lambda_0 + \sum_{i=1}^{N-d} \alpha_i \lambda_i - \alpha_0 \end{pmatrix},$$

has a solution path starting at $(y, \alpha_0, 0, \ldots, 0)$ which is trackable *on $(0, 1]$ and converges at $t \to 0$ with endpoint (x^*, λ^*), $x^* \in \mathcal{V}_{\mathbb{R}}(f)$.*

In some applications, the polynomial constraints which arise not only entail equalities of the form $f(x) = 0$, but also involve polynomial inequalities of the form $f(x) \geq 0$.

Definition 2.14. Given two polynomial systems with real coefficients $f : \mathbb{R}^N \to \mathbb{R}^n$ and $g : \mathbb{R}^N \to \mathbb{R}^m$, the set of points $\{x \in \mathbb{R}^N \mid f(x) = 0, g_1(x), \ldots, g_m(x) \geq 0\}$ is a *semialgebraic set.*

The homotopy continuation approach to the minimum distance problem can be extended to finding points on semialgebraic sets by ignoring the inequalities defining the set when setting up the distance problem. Solutions to the problem derived this way are clearly a lower bound for the minimum distance problem on the whole semialgebraic set, since they are solutions to the same optimization problem but with fewer constraints. A real critical point obtained from the homotopies can be evaluated on the inequality constraints to determine whether or not it actually belongs to the semialgebraic set.

This approach to the minimum distance problem is computationally feasible on modern hardware. `Bertini` is a robust package for performing a large number of NAG tasks, including homotopy continuation. Though it is powerful, it requires careful configuration to solve the minimum distance problem, as well as user selection of a large number of technical internal parameters. For larger examples like some of those presented in Chapter 4, calculating and formatting the polynomial systems in Theorems 2.12 and 2.13 in a format `Bertini` can process is a task requiring its own set of scripts, since the examples are far too large to manipulate by hand. `Bertini`'s interface is also entirely through file input and output rather than internal library calls, which requires additional work to integrate it into a larger program stack.

2.2 Topological data analysis

The motivating format for the data considered in TDA is a finite set of points in a metric space, particularly Euclidean space. Data in this form is also sometimes called a "point cloud". Given a point cloud as input, the TDA paradigm views the data as a discrete sampling of points from an underlying compact topological space containing infinitely many points. Our goal is to estimate topological features of the underlying space (roughly speaking, the number of holes in the space) from the sample.

2.2.1 Homology groups

The primary topological invariants we look to extract from point cloud samples are homology groups with coefficients in a field. Homology serves as one of the primary tools in the study of algebraic topology, for which [15] is a general introductory reference. In high level category theoretic terms, (singular) homology in a dimension $p \geq 0$ with coefficients in a ring R is a functor $H_p(-; R)$ from the category of topological spaces with continuous maps (**Top**) to the category of R-modules with R-module homomorphisms. Singular homology with arbitrary ring coefficients is too broad of a context for computation. For TDA we instead restrict our focus to (finite) simplicial homology with coefficients in the field $\mathbb{Z}/2$,

which is a variant of homology defined for finite simplicial complexes. Simplicial complexes are spaces built from simplices (points, lines, triangles, and their higher dimensional equivalents).

Definition 2.15. An *abstract simplicial complex* is a finite set S of non-empty subsets of $\mathbb{N}$ such that $s \in S$ implies that every subset of s is an element in S. If S is an abstract simplicial complex the elements of the set $V(S) = \cup_{s \in S} s$ are the *vertices* of S. The *dimension* of S is one less than the size of the largest set in S.

Since abstract simplicial complexes are the only complexes we will discuss, we drop the "abstract" going forward. We will also assume that the vertex set of a complex S with l vertices is $\{1, \ldots, l\}$ for convenience. Connecting simplicial complexes and topological spaces requires viewing the vertices of a simplicial complex as the vertices of a point, line, triangle, tetrahedron, or higher dimensional equivalent. Recall that the points $X = x_0, \ldots, x_l \subseteq \mathbb{R}^m$ are *affinely independent* if the vectors $x_1 - x_0, \ldots, x_l - x_0$ are linearly independent, and that the *convex hull* of X is the set of elements of the form $\sum_{i=0}^{l} a_i x_i$ where the a_i are nonnegative real numbers such that $\sum_{i=0}^{l} a_i = 1$.

Definition 2.16. Let S be a simplicial complex with vertices $\{1, \ldots, l\}$ and $x_1, \ldots, x_l \subseteq \mathbb{R}^m$ be an affinely independent set of points in $\mathbb{R}^m$. A *geometric realization* of S contains all points which are in the convex hull of $x_{s_1}, \ldots, x_{s_n}$ for any set $\{s_1, \ldots, s_n\} \in S$. A topological space is *triangulable* if it is homeomorphic to the geometric realization of some simplicial complex.

It can be shown that any two geometric realizations of the same simplicial complex are homeomorphic, so we will often refer to *the* geometric realization of a complex and denote it $|S|$. The computational advantage of using simplicial complexes is that they can be expressed by a finite amount of information. Homology identifies voids using a clever encoding of the simplices.

Definition 2.17. Let S be a simplicial complex. Then if S_i has elements the sets in S with size $i+1$, the *i'th chain group of S, $C_i(S)$,* is the set of finite formal sums of the form $\sum_{j=1}^{n} b_j s_j$ where $s_j \in S_i$ and $b_j \in \mathbb{Z}/2$ for all j. We set $C_{-1}(S) = 0$.

The p'th boundary operator is a map $\partial_p : C_p(S) \to C_{p-1}(S)$ which is defined as follows. We define ∂_0 to be the zero map $\partial_0 : C_0(S) \to 0$. For $p > 0$ and any basis element $s \in C_p(S)$, define $\partial_p(s) = \sum_{\{s' \in S_{p-1} | s' \subseteq s\}} s'$. ∂_p can be extended linearly from this definition on the basis elements of $C_p(S)$ to a homomorphism on the entirety of $C_p(S)$. Elements in the kernel of ∂_p are called *cycles* and this group is denoted $\ker(\partial_p) = Z_p(S)$. Elements in the image of ∂_{p+1} are called *boundaries* and this group is denoted $\mathrm{Im}(\partial_{p+1}) = B_p(S)$.

It can be shown that $\partial_p \circ \partial_{p+1} = 0$ for all $p \geq 0$, so that $B_p \subseteq Z_p$. The *p-th homology group* of S, $H_p(S)$, is the quotient group $Z_p(S)/B_p(S)$. $H_p(S)$ is a finite dimensional vector space, and its rank is called the *p-th Betti number* of S, $\beta_p(S)$.

While the construction of homology groups is quite formal, the elements of the homology groups informally represent loops and higher dimensional equivalents in a space. A basis element of the 0'th homology group of a complex S represents a single connected component of S's geometric realization, a basis element of $H_1(S)$ represents a set of loops which can all be deformed within the space S into a loop which encloses a 2D void, and basis elements of $H_2(S)$ count 3D voids.

Definition 2.18. Let S and T be simplicial complexes. A map $f : V(S) \to V(T)$ is a *simplicial map* if $\sigma \in S$ implies that $f(\sigma) \in T$. Suppose that $|S|$ and $|T|$ are geometric realizations of S and T on vertex points $x_1, \ldots, x_m$ and $y_1, \ldots, y_n$ respectively. Then the geometric realization of f is a function $|f| : |S| \to |T|$ defined by $|f|(\sum_{i=1}^m a_i x_i) = \sum_{i=1}^m a_i y_{f(i)}$.

Really we are interested in topological spaces, and simplicial complexes are just useful ways to encode spaces. The last step in cementing this connection is to show that not only can spaces be represented by complexes, but that continuous maps can be represented by simplicial maps as well. Recall that two continuous functions $f, g : X \to Y$ are *homotopic* if there exists a continuous function $H : X \times [0,1] \to Y$ where $f(x) = H(x,0)$ and $g(x) = H(x,1)$ for all $x \in X$. This is written $f \simeq g$. If there exist continuous $h : X \to Y$ and $k : Y \to X$ such that $h \circ k \simeq 1_Y$ and $k \circ h \simeq 1_X$ then X and Y are *homotopy equivalent*. Up to homotopy, simplicial approximations to continuous functions suffice, and homology is homotopy invariant.

Theorem 2.19 (Simplicial Approximation Theorem)**.** *Let X and Y be triangulable spaces and $g : X \to Y$ a continuous map between them. There exist simplicial complexes S and T and a simplicial map $f : V(S) \to V(T)$ where X and Y are homeomorphic to $|S|$ and $|T|$ respectively. Furthermore, $|f| \simeq g'$, where $g' : |S| \to |T|$ is the composition of maps $|S| \to X \to Y \to |T|$, the middle function from X to Y is g, and the other functions are homeomorphisms.*

Theorem 2.20. *For any simplicial complexes S and T, simplicial map $h : V(S) \to V(T)$, and $p \geq 0$ there exists a $\mathbb{Z}/2$-linear map $H_p(h) : H_p(S) \to H_p(T)$. Furthermore, if U is also a simplicial complex and both $f : V(S) \to V(T)$ and $g : V(T) \to V(U)$ are simplicial maps, the map $H_p(h \circ f)$ is equal to the composition $H_p(h) \circ H_p(f)$. If $1_S : V(S) \to V(S)$ is the identity map, then $H_p(1_S)$ is the identity map on $H_p(S)$. If $j : V(S) \to V(T)$ is a simplicial map and $|f| \simeq |j|$, then $H_p(j) = H_p(f)$.*

These theorems establish a few points. A direct corollary is that the homology groups of two homotopy equivalent triangulable spaces are isomorphic, and that different simplicial complexes with homeomorphic geometric realizations have isomorphic fundamental groups. For triangulable spaces, then, the homology groups are independent of the triangulation. We will assume all the spaces discussed are triangulable and for a triangulable space X, $H_p(X)$ is the homology group for some underlying complex of X. The second theorem shows that H_p is a functor from the category of finite simplicial complexes with simplicial maps to the category of finite dimensional $\mathbb{Z}/2$-vector spaces with linear maps. Combining the two theorems, it follows that any continuous function between triangulable spaces induces a unique homomorphism (up to isomorphism) between their homology groups.

2.2.2 Building simplicial complexes from data

The first step in applying homology to point cloud data is transforming the data into an interesting topological space. Let X be a finite subset of $\mathbb{R}^n$. The topology X directly inherits from the standard topology on $\mathbb{R}^n$ is the discrete topology, which contains no additional information. A simple way to give X more structure is to "thicken" the point cloud by choosing a parameter $\epsilon > 0$ and replacing X

with the space $\cup_{x \in X} \bar{B}_\epsilon(x)$ consisting of closed balls with radius ϵ centered at the points of X. This space admits a rather simple triangulation.

Definition 2.21. Let $\mathcal{U} = \{U_i\}_{i=1}^l$ be a finite collection of open sets. The *nerve* or *Čech complex* of the collection $\mathcal{U}$ is the simplicial complex $C(\mathcal{U})$ on vertices $1, \ldots, l$ which contains a subset $\{s_1, \ldots, s_r\} \subseteq \{1, \ldots, l\}$ if the intersection $U_{s_1} \cap \cdots \cap U_{s_r}$ is nonempty. If X is a finite set of points, then $C_\epsilon(X)$ denotes the nerve of the space $\cup_{x \in X} \bar{B}_{\epsilon/2}(x)$.

If Y is a subspace of $\mathbb{R}^n$ and $\mathcal{U}$ is a finite cover of Y where the nonempty intersection of any finite set of elements in $\mathcal{U}$ is contractible, the Nerve Theorem (4G.3 [15]) shows that $|C(\mathcal{U})|$ is homotopy equivalent to Y. The thickening of X by balls of radius $\frac{\epsilon}{2}$ is thus always equivalent to $C_\epsilon(X)$ for the purpose of computing homology. It might seem tempting to apply this to our example X by trying to ascertain what parameter ϵ best recovers the underlying space X is sampled from. TDA methods instead consider how the homology of $C_\epsilon(X)$ changes as the parameter ϵ varies.

Although the nerve built from thickening X is useful in a theoretical context, actually constructing and storing the complex $C_\epsilon(X)$ can be computationally impractical as it depends very precisely on the various distances between points. A more computationally tractable construction is the *Rips complex*.

Definition 2.22. Let X be a finite subset of $\mathbb{R}^n$ containing l points and $\epsilon > 0$. The *Rips complex* of X corresponding to ϵ is a simplicial complex $R_\epsilon(X)$ with vertices $\{1, \ldots, l\}$ such that a set $\{s_1, \ldots, s_u\} \subseteq X$ is an element of $R_\epsilon(X)$ if the points $\{x_{s_1}, \ldots, x_{s_u}\}$ are pairwise less than distance ϵ apart.

Constructing Rips complexes only requires computing and storing information about the distances between pairs of points in X. The drawback is that no result like the Nerve Theorem exists for Rips complexes, so it is unclear whether $R_\epsilon(X)$ can faithfully extract information from X about the space from which we are viewing X to be sampled. The two constructions are not entirely unrelated, however.

Theorem 2.23 (De Silva and Ghrist [12]). *If X is a finite set of points in $\mathbb{R}^d$ and $\epsilon > 0$ there is a chain of inclusions*

$$R_{\epsilon'}(X) \subseteq C_\epsilon(X) \subseteq R_\epsilon(X)$$

whenever $\frac{\epsilon}{\epsilon'} \geq \sqrt{\frac{2d}{d+1}}$.

Composing the above inclusions with the homology functor H_p yields inclusions from $H_p(R_{\epsilon'}(X)) \to H_p(C_\epsilon(X)) \to H_p(R_\epsilon(X))$. This shows that if an element of $H_p(R_{\epsilon'}(X))$ has a non-zero inclusion into $H_p(R_\epsilon(X))$ (the void corresponding to that element was not filled in after expanding ϵ' to ϵ), that feature must also be an element of $C_\epsilon(X)$.

2.2.3 Persistent homology

As mentioned in the previous section, the key maneuver in TDA is to consider all of the homology groups $H_p(C_\epsilon(X))$ simultaneously, rather than try to construct a faithful representation of the underlying space from a single parameter value. Persistent homology is an algebraic theory describing how to track homology elements as the parameter value changes. This theory is presented in [7] and [13] for instance, and expanded into a more general categorical framework in [6]. If $\epsilon' \geq \epsilon$ then there is an inclusion function $i : C_\epsilon(X) \to C_{\epsilon'}(X)$. Correspondingly, we can consider the homology group $H_p(i(C_\epsilon(X))$ to begin extracting information about how the homology has changed with changes to the parameter.

Definition 2.24. Given a sequence of spaces with continuous maps between them $X_0 \to X_1 \to \cdots \to X_m \to X_{m+1} \to \ldots$, there is a corresponding sequence of homology groups and homomorphisms $f_i : H_p(X_i) \to H_p(X_{i+1})$, $H_p(X_0) \to H_p(X_1) \to \ldots H_p(X_m) \to H_p(X_{m+1}) \to \ldots$. If the function $f_i^j : H_p(X_i) \to H_p(X_j)$ is the composition $f_{j-1} \circ f_{j-2} \circ \cdots \circ f_{i+1} \circ f_i$, the (i, j)-*persistent homology group* is the image of the map f_i^j.

To unify different ways of producing sequences of nested spaces into a single framework, we can consider *filter functions* on a space Y. A filter function is a function $f : Y \to \mathbb{R}$, and for a real value $a \in \mathbb{R}$ the *sublevel set* associated with a is $f^{-1}(-\infty, a]$. If $b > a$, notice that $f^{-1}(-\infty, a] \subseteq f^{-1}(\infty, b]$, and that this subset

relation implies the existence of an inclusion function. For the finite point sample $X \subseteq \mathbb{R}^n$ the function $d_X : \mathbb{R}^n \to \mathbb{R}^{\geq 0}$ defined by $d_X(y) = \inf_{x \in X} \|x - y\|$ is a continuous filter function. The sublevel set $d_X^{-1}(-\infty, \frac{\epsilon}{2}]$ is precisely the thickening of X by $\frac{\epsilon}{2}$ balls. By the Nerve Theorem, the simplicial homology groups of the form $H_p(d_X^{-1}(-\infty, \epsilon])$ are the homoloy groups $H_p(C_\epsilon(X))$. We will assume that for all the spaces and filter functions considered that the sublevel sets are triangulable.

The definition of persistent homology groups makes sense when we have an at most countable sequence of spaces and maps, but the motivating sequence of spaces $C_\epsilon(X)$ for all $\epsilon \geq 0$ do not necessarily fit this criterion. We require the following restriction. Recall that the partially ordered set $(\mathbb{R}, \leq)$ can be viewed as a category with objects the elements of $\mathbb{R}$ and a single arrow $x \to y$ if $x \leq y$. The arrow for $x \leq x$ is the identity arrow.

Definition 2.25 ([9])**.** Let J be a functor from $(\mathbb{R}, \leq)$ to the category of finite dimensional vector spaces over $\mathbb{Z}/2$ (**FDVec**). A point $a \in \mathbb{R}$ is a *homological critical value* of J if, for all sufficiently small $\epsilon > 0$, the arrow $J(a - \epsilon \leq a + \epsilon)$ is not an isomorphism. The functor J is *tame* if it has finitely many homological critical values.

Given a topological space Y and filter function $g : Y \to \mathbb{R}$, there is a corresponding functor G from $(\mathbb{R}, \leq)$ to the category of triangulable spaces with continuous maps given by $a \mapsto g^{-1}(-\infty, a]$ and $a \leq b \mapsto i : G(a) \to G(b)$, the inclusion map. The function g is tame if $H_p \circ G$ is tame for all $p \geq 0$.

Suppose F is a tame functor from $(\mathbb{R}, \leq)$ to **FDVec**, and F's homological critical values are $a_0 < a_1 < \cdots < a_n$. We can identify F with a finite sequence of vector spaces with maps between them (a functor from the poset $(\mathbb{N}, \leq)$ to **FDVec**) by using an interleaving sequence of real numbers $b_0 < a_0 < b_1 < a_1 < b_2 < \cdots < a_n < b_{n+1}$. The corresponding sequence is $F(b_0) \to F(a_0) \to F(b_1) \to F(a_1) \to \cdots \to F(a_n) \to F(b_{n+1})$ where the maps from two elements $F(q)$ and $F(r)$ for $q < r$ in this sequence are defined by $F(q \leq r)$. The (c, d)-persistent homology group for F can then be defined as the image of $F(b_i \leq b_j)$ where $a_{i-1} \leq c, b_i < a_i$ and similarly $a_{j-1} < b_j, d \leq a_j$.

Let $X_0 \to X_1 \to \ldots$ be a sequence of spaces with continuous maps between them. From an intuitive perspective, there are three types of "events" which

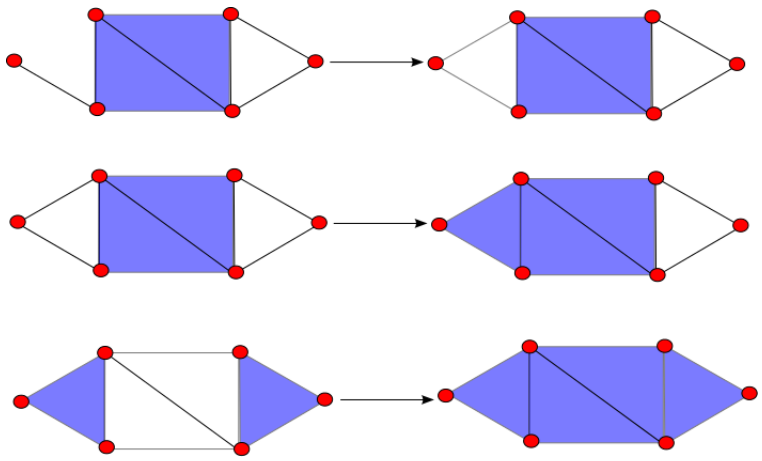


Figure 2.2: Different events in the life of a homology feature. In the top row a feature is born, in the middle two features merge, and in the bottom a feature dies.

can change the status of a particular homology feature throughout a sequence of inclusions. A feature $[\sigma_d] \in H_p(X_d)$ can be "born" at a point d when it is not a member of the (c, d)-persistent homology group for any $c < d$. It can "merge" with a different homology feature at a point d if $f_c^d([\sigma_c]) = f_{c'}^d([\tau_{c'}])$ for some $[\tau_{c'}] \in H_p(X_{c'})$ and $f_c^{d-1}([\sigma_c]) \neq f_{c'}^{d-1}([\tau_{c'}])$. Finally, the feature $[\sigma_c]$ can "die" at a point d if $f_c^d([\sigma_c]) = 0$ and $f_c^{d-1}([\sigma_c]) \neq 0$. This is shown geometrically in Figure 2.2. These notions can be formalized in an algebraic structure known as a *persistence module*.

Definition 2.26. Let F be a functor from $(\mathbb{N}, \leq)$ to **FDVec**. The *persistence module* associated with this sequence is a $\frac{\mathbb{Z}}{2}[x]$ module. Its elements are members of the direct sum group $\bigoplus_{i\in\mathbb{N}} F(i)$. The element $x \in \frac{\mathbb{Z}}{2}[x]$ acts via a shift map, that is $x \cdot [\sigma_i] = F(i \leq i+1)([\sigma_i]) \in F(i+1)$ for $[\sigma_i] \in F(i)$, an element b in $\mathbb{Z}/2$ acts by via its vector space action, and this action extends linearly to general elements of $\frac{\mathbb{Z}}{2}[x]$ and $\bigoplus_{i\in\mathbb{N}} F(i)$.

Persistence modules are *graded* $\frac{\mathbb{Z}}{2}[x]$ modules because, if $R_n = \frac{\mathbb{Z}}{2}/\langle x^{n+1}\rangle$ is the ring of polynomials up to degree n, $R_n \cdot F(i)$ is contained in $F(i+n)$. Since $\mathbb{Z}/2$ is a field, the structure theorem for modules over a principal ideal domains applies. It combines with the graded structure on a persistence module for the following

result. We note here that our restriction to finite simplicial complexes causes the $F(i)$ to be finite dimensional vector spaces, which is a technical requirement for the following theorem.

Theorem 2.27 (Fundamental Theorem of Persistent Homology [29])**.** *Let $\mathcal{M}$ be a persistence module over $\frac{\mathbb{Z}}{2}[x]$ built from a tame functor $F : (\mathbb{N}, \leq) \to$ **FDVec**. Then:*

$$\mathcal{M} \cong \bigoplus_{i \in I} x^{t_i} \cdot \frac{\mathbb{Z}}{2}[x] \oplus \left(\bigoplus_{j \in J} x^{r_j} \cdot (\frac{\mathbb{Z}}{2}[x]/(x^{s_j} \cdot \frac{\mathbb{Z}}{2}[x])) \right)$$

for some index sets I and J and naturals t_i, r_j, and s_j.

Theorem 2.27 tells us that the persistence module $\mathcal{M}$ is exactly defined by features (generators of the summands in the theorem) that are born at sequence index t_i and do not die, and features that are born at sequence index r_j and die at sequence index $r_j + s_j$. In the case of a sequence constructed from a tame functor on $\mathbb{R}$, these sequence indices can be translated back into the real numbers corresponding to an interleaving sequence and so give information about the persistent homology groups of the entire functor. There are two standard ways to encode the information contained in a tame functor, which are mathematically equivalent. The proof of the following theorem relies on Theorem 2.27.

Theorem 2.28 ([6])**.** *Let F be a tame functor from $(\mathbb{R}, \leq)$ to **FDVec**. There exists a* barcode *$\mathcal{B}$ associated with F, which is a multiset of intervals of the form $[a, \infty)$ and $[a, b)$. For real numbers $c < d$, the rank of the (c, d)-persistent homology group of F is the number of intervals in $\mathcal{B}$ which intersect the interval $[c, d]$. The* persistence diagram *D of F is a multiset of points in $\bar{\mathbb{R}} \times \bar{\mathbb{R}}$ where $\bar{\mathbb{R}} = \mathbb{R} \cup \{-\infty, \infty\}$. Elements of D are pairs (a, b) where $[a, b)$ is an element of $\mathcal{B}$, along with points (x, x) counted with infinite multiplicity for all $x \in \mathbb{R}$. Viewing D as a subset of the extended plane $\bar{\mathbb{R}} \times \bar{\mathbb{R}}$, the rank of the (c, d)-persistent homology group for F is the number of points of D (counted with multiplicity) in the upper-left box with lower-right corner (c, d) in $\bar{\mathbb{R}}^2$.*

These encodings are quite similar, but admit somewhat different visualizations. We display a barcode as a collection of horizontal lines plotted against a horizontal parameter axis, while a persistence diagram can be visualized as a collection of

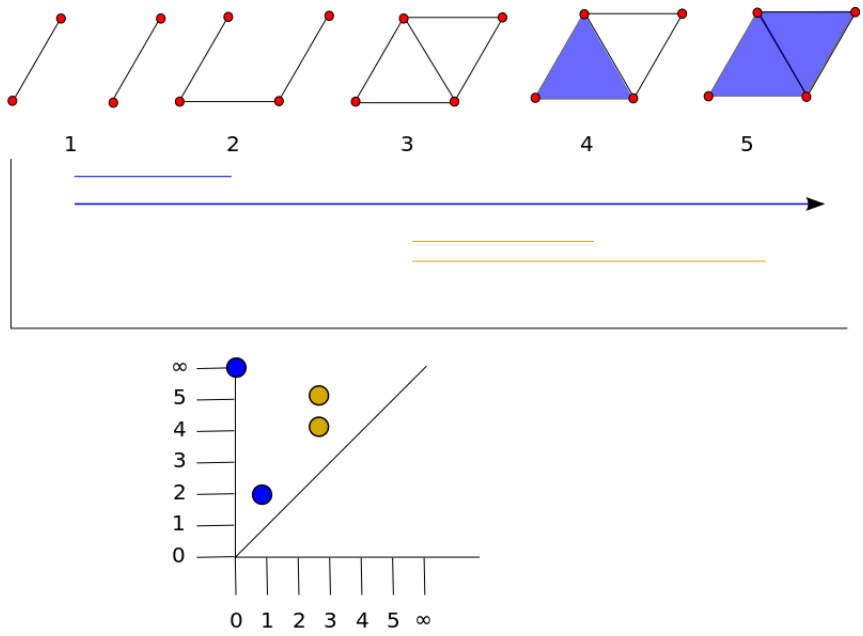


Figure 2.3: The persistence diagram and barcode of a filtered complex. The top figure is the complex as it changes with parameter values. The bottom two figures are the corresponding functor's persistence diagram and barcode. Blue bars and points represent 0 dimensional homology, whereas yellow bars and points represent 1 dimensional homology. An arrow on a bar indicates that the homology feature corresponding to the bar lives forever.

points in the extended plane. As always, the motivating example is a tame functor $H_p(C_\epsilon(X))$. Qualitatively, we regard long bars in the barcode for $H_p(C_\epsilon(X))$ as relevant features which exist in the underlying space from which X is sampled. Shorter bars are features which are born and then die quickly, and can be taken as noise. The next section makes this more precise.

2.2.4 Persistence diagram metrics and the stability theorem

Suppose that we start with a finite point sample $X \subseteq \mathbb{R}^n$ as in the previous sections, and then form a perturbed sample X' by moving some of the points away from their original positions by a small distance. What we would like to show is

that in situations like this the persistence diagrams corresponding to the functors $H_p(C_\epsilon(X))$ and $H_p(C_\epsilon(X'))$ are not very different. The first step in making this precise is defining a measure of distance between diagrams.

Definition 2.29. For any two points $(a_1, b_1), (a_2, b_2) \in \bar{\mathbb{R}} \times \bar{\mathbb{R}}$, the ℓ_∞ norm between the points is $\ell_\infty((a_1, b_1), (a_2, b_2)) = \max(|a_2 - a_1|, |b_2 - b_1|)$. The value $\infty - \infty$ is taken to be 0 and $\infty - a = \infty$, for any $a \neq \infty$.

Definition 2.30. For any two bounded subsets C and D of a metric space M with metric d, the *Hausdorff distance* between A and B, $d_H(A, B)$, is the maximum of $\sup_{a \in A} \inf_{b \in B} d(a, b)$ and $\sup_{b \in B} \inf_{a \in A} d(a, b)$.
Let D_1 and D_2 be two persistence diagrams, and let B be the set of bijections between D_1 and D_2. The *bottleneck distance* $d_B(D_1, D_2)$ between D_1 and D_2 is $\inf_{b \in B} \sup_{p \in D_1} \ell_\infty(p, b(p))$.

One way to motivate this definition is to consider the points in D_1 as workers, the points in D_2 as jobs to which a single worker can be assigned, and $\ell_\infty(p, b(p))$ as the amount of time it takes for worker p to do job $b(p)$. The bottleneck distance is the amount of time it takes to complete all the jobs for the most efficient matching of workers to jobs. There is always some bijection in B since each persistence diagram includes the set of diagonal points (x, x) counted with infinite multiplicity. It can additionally be checked that d_B is an extended metric on the set of persistence diagrams with finitely many off-diagonal points (the distance between two diagrams is potentially infinite).

Theorem 2.31 (The Stability Theorem, [9, 8])**.** *Suppose that $f, g : Y \to \mathbb{R}$ are tame filter functions on the topological space Y with corresponding sublevel functors F and G. If $\|f - g\|_\infty = \sup_{y \in Y} |f(x) - g(x)|$ is the L_∞ norm and D_f and D_g are persistence diagrams corresponding to H_pF and H_pG respectively, then $d_B(D_f, D_g) \leq \|f - g\|_\infty$.*

This theorem covers our motivating example. If we perturb points in a sample X by at most δ, then $\|d_X - d_{X'}\|_\infty \leq \delta$. By the Stability Theorem, if D_1 and D_2 are diagrams corresponding to $H_p(C_\epsilon(X))$ and $H_p(C_\epsilon(X'))$ then $d_B(D_1, D_2) \leq \delta$. The stability theorem covers more than this, however. Given two finite point samples

X and X' with $d_H(X, X') \leq \delta$, $\|d_X - d_{X'}\|_\infty$ still holds and $d_B(D_1, D_2) \leq \delta$. We can even use the Stability Theorem to show that the theory of persistent homology recovers homology information about an underlying space from a finite point cloud sample.

Definition 2.32. Let $X \subseteq \mathbb{R}^n$ be a bounded subspace of $\mathbb{R}^n$. A finite set of points $X_0 \subseteq \mathbb{R}^n$ is an ϵ-sample of X if $d_H(X_0, X) \leq \epsilon$. If $d_X : \mathbb{R}^n \to \mathbb{R}$ is the distance function for X, the *homological feature size* of X, $\text{hfs}(X)$, is the smallest non-zero homological critical value of d_X.

Theorem 2.33 (Homology Inference Theorem, [9])**.** *Suppose $X \subseteq \mathbb{R}^n$ is a bounded subspace of $\mathbb{R}^n$ with tame distance function d_X, and that $hfs(X) > 4\delta$ for some $\delta > 0$. Then if X_0 is a δ-sampling of X, the number of points (counted with multiplicity) which are above and to the left of the point $(\delta, 3\delta)$ in the persistence diagram corresponding to $H_p(C_\epsilon(X_0))$ is the Betti number $\beta_k(X)$.*

Corollary 2.34. *Suppose $X \subseteq \mathbb{R}^n$ is a bounded subspace of $\mathbb{R}^n$ with tame distance function d_X, and that $hfs(X) > 4\delta$ for some $\delta > 0$. If X_0 is a δ-sampling of X, then the Betti number $\beta_k(X)$ is at least as large as the number of points (counted with multiplicity) which are above and to the left of the point $(\delta\sqrt{\frac{n+1}{2n}}, 3\delta)$ in the persistence diagram corresponding to $H_p R_\epsilon(X_0)$.*

Proof. Let $\delta' = \delta\sqrt{\frac{n+1}{2n}}$. By Theorem 2.28 we know that the number of points above and to the left of $(\delta', 3\delta)$ in the persistence diagram for $H_p(R_\epsilon(X_0))$ is the rank of the $(\delta', 3\delta)$-persistent homology group for this functor, call it G_R. By the Homology Inference Theorem it suffices to show that the rank of the $(\delta, 3\delta)$-persistent homology group of the functor $H_p(C_\epsilon(X_0))$ is at least the rank of G_R. From Theorem 2.23 there exist inclusion maps from $R_{\delta'}(X_0) \to C_\delta(X_0) \to C_{3\delta}(X_0) \to R_{3\delta}(X_0)$. Applying the homology functor H_p we get the sequence of finite dimensional vector spaces with linear maps $H_p(R_{\delta'}(X_0)) \to H_p(C_\delta(X_0)) \to H_p(C_{3\delta}(X_0)) \to H_p(R_{3\delta}(X_0))$. Since the induced inclusion function from $H_p(R_{\delta'}(X_0))$ to $H_p(R_{3\delta}(X_0))$ is equal to the composition of all the functions in this diagram, we see that the rank of the image of the inclusion $H_p(C_\delta(X_0)) \to H_p(C_{3\delta}(X_0))$ is an upper bound on the rank of the image of the inclusion $H_p(R_{\delta'}(X_0)) \to H_p(R_{3\delta}(X_0))$ as desired. □

2.2.5 Software implementations

Software for calculating the persistent homology of a point cloud must perform two steps corresponding to the previous two sections. It must be able to represent, store, and build simplicial complexes from the input points at different parameter values. Ultimately this requires storage of information for each simplex in the complex for the largest parameter value analyzed.

Even when limiting the dimension of the simplices to 3 so that 3D voids can be identified, the total number of simplices is in the worst case more than $\binom{n}{4}$ where n is the number of points in the point cloud. For even 510 input points this is approximately 2.8 billion simplices to track, and the memory requirements can quickly skyrocket into hundreds of gigabytes even when utilizing specialized data structures for sparse matrices.

Calculating a persistence diagram from a filtration is made possible by use of an algorithm for this purpose is called the "standard algorithm", which was first presented for calculating persistent homology over arbitrary fields in [29]. The computational complexity is cubic in the number of simplices. Correspondingly, keeping down the number of points in a point cloud which still represents a sampling from a space of interest is a practical necessity for persistent homology computations.

The article [23] provides detailed benchmarking results for the current software implementations of persistent homology. Based on its recommendations, the package `DIPHA` [3] is used for computations in Chapter 3. `DIPHA` operates through its own specialized file input and output formats, so it is not particularly user friendly. Additionally, it provides no feedback as to the state of a persistent homology computation while it is running. Integrating `DIPHA` into a data analysis stack with NAG necessitates tying together many scripts provided by `DIPHA` as well as code tailored to the various stack components.

Chapter 3

Results I: Sampling Algorithm

Bounded portions of real algebraic varieties/real semialgebraic sets fulfill the TDA paradigm, since a real algebraic variety X is a closed subspace of some high dimensional Euclidean space $\mathbb{R}^N$. Given an ϵ-sampling X_0 of the compact space $X \cap \bar{B}_M(0)$, we could use this sampling as input to a persistent homology computation and attempt to identify some of the homological features of $X \cap \bar{B}_M(0)$. A sampling algorithm which returns a provably ϵ-dense sample accomplishes two main goals. The algorithm must both compute sample points, and it must be able to verify that the set of points it has computed is ϵ-dense. On the one hand, applying the Homology Inference Theorem and its corollaries to identify features in the underlying space gets easier as ϵ shrinks. On the other, it is not enough to obtain any arbitrary ϵ-sampling. Since TDA methods require significant computational resources that scale up rapidly with increasing numbers of points in the sample, it is necessary to find provably ϵ-dense samplings which take care to minimize the number of points in the sample.

Techniques for determining ϵ-samplings of real semialgebraic sets have been developed. The methods in [25], for instance, determine a representation of polynomial systems in the Bernstein basis, and combine properties of that representation with nonlinear programs to obtain a sampling. This strategy returns points that are in the worst case distance ϵ away from the variety. The method also does not minimize the size of the sample returned.

This Chapter presents a new sampling algorithm (SA) based on using numerical algebraic geometry to solve the minimum distance problem for a real algebraic

variety. The algorithm is more flexible than previous methods in how it can produce a provably ϵ-dense sampling. Furthermore, every point in the sample is very close to some point on the variety. A version of the algorithm has been implemented as well as heuristic techniques for reducing the number of points in the sample. The chapter also provides relevant implementation details.

3.1 Assumptions, input, and output

We first consider a variant of the SA which applies to pure dimensional real algebraic varieties, and then detail a modification which extends it to semialgebraic sets with components of mixed dimension. Throughout this chapter we will use the terms "box" and "rectangle" to refer to a subset of Euclidean space which has the form $\Pi_{i=1}^{N}[c_i, d_i]$. Unless otherwise noted, boxes and rectangles are not necessarily hypercubes. That is to say they they can have a different length in each dimension. The *measure* of a box $\Pi_{i=1}^{N}[c_i, d_i]$ refers to the value $\Pi_{i=1}^{N}(d_i - c_i)$.

Recall from Chapter 2 that a polynomial system must meet some conditions to employ homotopy continuation to solve the minimum distance problem on the polynomial system's variety. The condition are that a variety of dimension d in $\mathbb{R}^N$ must be defined by a polynomial system having N unknowns, $N-d$ equations, and real coefficients. Chapter 2 details how an arbitrary system can be transformed into a system with the desired number of equations, so we will assume that this transformation has been done. Some extra steps are necessary if the system has been reduced in this way, which we will discuss briefly along with the algorithm's extension.

- **Input:** A polynomial system $f : \mathbb{C}^N \to \mathbb{C}^{N-d}$ of polynomials $f_1, \ldots, f_{N-d}$ with real coefficients such that $\mathcal{V}(f)$ is a pure d-dimensional variety, the *overall bounds* $R = \Pi_{i=1}^{N}[a_i, b_i] \subseteq \mathbb{R}^N$ to search, a density parameter $\epsilon > 0$, and a precision parameter $0 < \delta \leq \epsilon$.

- **Output:** A finite ϵ-sampling of $\mathcal{V}_{\mathbb{R}}(f) \cap R$ such that the distance from any point in the sample to a point on $\mathcal{V}_{\mathbb{R}}(f)$ is at most δ.

We will say that the function MinPoint(y) returns (a very close estimate of) a point $x \in \mathcal{V}_{\mathbb{R}}(f)$ that minimizes the distance from y to $\mathcal{V}_{\mathbb{R}}(f)$ for generic $y \in \mathbb{R}^N$ by applying homotopy continuation with the homotopy from Theorem 2.12, and also returns any other real critical values calculated during the homotopy continuation. Similarly, the function CritPoint(y) returns some real critical point $x \in \mathcal{V}_{\mathbb{R}}(f)$ such that $y - x$ is perpendicular to the tangent space of $\mathcal{V}_{\mathbb{R}}(f)$ at x. The homotopy continuation for these functions can be configured to return points which have distance at most δ from $\mathcal{V}_{\mathbb{R}}(f)$. For complete theoretical correctness, the rounding parameter δ should be subtracted from the minimum distance obtained from MinPoint(y) since all values relying on homotopy continuation are estimates with precision δ. In practice, δ is no larger than 10^{-7}. The implemented version of the algorithm supports this adjustment.

3.2 Maintaining and checking the search space

The key data structure underpinning the algorithm is called the *search space*. The search space maintains information about sample points on $\mathcal{V}_{\mathbb{R}}(f)$ that have been identified and about regions of space that do not contain any points on $\mathcal{V}_{\mathbb{R}}(f)$. For the sake of computability, these regions are rectangular. Furthermore, the search space provides a mechanism by which any rectangular region $R' \subseteq \mathbb{R}^N$ can be checked against the currently stored information to see if the sample points already obtained are a provably ϵ-dense covering of $\mathcal{V}_{\mathbb{R}}(f) \cap R'$. The overall bounds R are the main argument to check this way, but checking other regions also proves useful. Finally, the search space can produce on demand a point $y \in R$ which, according to the information about sample points and regions of exclusion the search space is storing, is most likely to be as far away from $\mathcal{V}_{\mathbb{R}}(f)$ as possible while also belonging to a region of R which is not accounted for either by sample points or exclusion regions.

3.2.1 Search space interface and spatial databases

To store and access information about sampling points and regions of exclusion efficiently, the search space makes use of a "spatial database" data structure.

Spatial databases such as R^* trees [4] and k-d trees [5] allow for the storage and retrieval of data points which are individually indexed by a point or rectangular region in $\mathbb{R}^N$. Spatial databases support querying the data via an *intersection query*. Upon indicating a rectangular query region in $\mathbb{R}^N$, the database efficiently identifies and returns all those data points which have an index rectangle that intersects with the query region. These types of data structures find frequent use in clustering applications and geographic information systems. There is nontrivial computational overhead associated with inserting data points as well as performing an intersection query, but the overhead is small relative to other steps in the algorithm.

The search space data structure should implement the following five functions as an interface in any way that is computationally feasible. Note that it is not necessary for index boxes to be subboxes of the overall bounds, R. We will discuss implementing the interface assuming access to a spatial database DB which allows for intersection queries. If $R' \subseteq \mathbb{R}^N$ is a rectangular region, the function $\textsc{insert}(DB, \text{data}, R')$ denotes inserting data into the database DB with index region R'. Similarly, the function $\textsc{query}(DB, R')$ returns a set consisting of all pairs (d, R'') of data points paired with their index R'' in DB where $R' \cap R''$ is not empty.

1. $\textsc{InsertSamplePoint}(z)$. This function saves the point $z \in R$ as a sample point. When using the spatial database DB, this entails building the hypercube $A \subseteq B_\epsilon(z) \subseteq \mathbb{R}^N$ with center z and diagonal length 2ϵ, then performing $\textsc{insert}(DB, (z, \text{sample}), A)$. We call A a "sample box". If the input point z is not in R then it is not added.

2. $\textsc{InsertExclusionRegion}(z, \delta)$. This function builds the hypercube $A \subseteq B_\gamma(z) \subseteq \mathbb{R}^N$ with center z and diagonal length 2γ, then saves A as a region which contains no points in $\mathcal{V}_{\mathbb{R}}(f) \cap R$. We call A an "exclusion region" or "exclusion box". Using DB, this function performs $\textsc{insert}(DB, \text{exclusion}, A)$.

3. $\textsc{CheckRegion}(R')$. This function checks whether the previously saved sample points and exclusion regions prove that the currently obtained sample is

an ϵ-sample of $\mathcal{V}_{\mathbb{R}}(f) \cap R'$ for the rectangular region $R' \subseteq \mathbb{R}^N$. CheckRegion returns False if this is not the case, and True otherwise. The implementation of this function using DB is involved, and we reserve it for the next section.

4. ReturnTestPoint(). This function returns a point $z \in R$ which is not contained in any previously saved exclusion region, is not within ϵ of any previously saved sample point, and is far away from $\mathcal{V}_{\mathbb{R}}(f) \cap R$ subject to these restrictions. The implementation of this function is closely related to CheckRegion, and will also be discussed in the next section.

5. OutputSample(). This function returns the set of sample points which have previously been saved. Using DB, this returns the list query(DB, R) with the data points corresponding to exclusion boxes filtered out.

The function ReturnTestPoint is used to reduce the number of calls to MinPoint the overall algorithm has to make. MinPoint is computationally expensive relative to the information it returns, particularly considering that many of the critical values found by the homotopy continuation are not real. If a test point $y \in R$ is as far away as possible from $\mathcal{V}_{\mathbb{R}}(f)$, the call MinPoint$(y)$ allows us to save a relatively large exclusion region to the search space.

3.2.2 Checking regions

Given some sample points and exclusion regions have been stored in the spatial database DB, checking that the sample is an ϵ-sample of $\mathcal{V}_{\mathbb{R}}(f) \cap R'$ for some $R' \subseteq \mathbb{R}^N$ is equivalent to checking that R' is completely covered by the index boxes for the stored sample and exclusion regions. We record this as a Proposition.

Proposition 3.1. *Let $\mathcal{B} = \mathcal{R}_1 \cup \mathcal{R}_2$ be a collection of boxes in $\mathbb{R}^N$ such that $R' \subset \cup_{B \in \mathcal{B}} B$. Suppose the elements of $\mathcal{R}_1$ are boxes with center points less than distance δ away from $\mathcal{V}_{\mathbb{R}}(f)$ and with diagonals of length less than $2(\epsilon - \delta)$. Also suppose that $\mathcal{V}_{\mathbb{R}}(f) \cap B = \emptyset$ for all $B \in \mathcal{R}_2$. Then the collection of center points of boxes in $\mathcal{R}_1$ is an ϵ-sampling of $\mathcal{V}_{\mathbb{R}}(f) \cap R'$.*

Proof. Suppose to the contrary that there is a point $x \in \mathcal{V}_{\mathbb{R}}(f) \cap R'$ that is not within ϵ of the center point of any box in $\mathcal{R}_1$. Then $x \notin B$ for any $B \in \mathcal{R}_1$. Since $x \in R'$ it follows that $x \in B'$ for some $B' \in \mathcal{R}_2$, but this contradicts that $\mathcal{V}_{\mathbb{R}}(f) \cap B' = \emptyset$. □

If R' is entirely contained in one of the exclusion or sample boxes, Proposition 3.1 suggests a straightforward procedure for identifying that the current sampling saved in DB is an ϵ-sampling. The function QUERY(DB, R') returns a list of the data points and index rectangles in DB that intersect R', and so we loop through the list of index boxes until we find the one that contains R'. In most cases, however, R' will not be contained in a single index box. Faced with this situation, we can split R' into smaller boxes and repeat the procedure. For a box $B \subseteq \mathbb{R}^N$, let the function SPLITBOX(B) return a list of boxes which are properly contained in B and whose union is B. As a technical point, we must also require that repeated applications of SPLITBOX eventually produce a cover of B by smaller boxes where the length of every side of each smaller box is smaller than the smallest side length of B.

Algorithm 3.2 CHECKREGION

```
function CheckRegion(R')
    BoxList ← R'
    while BoxList ≠ ∅ do
        B ← last element of BoxList
        if query(DB, B) = ∅ then                    ▷ The box B not covered.
            return False
        else if A box in query(DB, B) contains B then
            BoxList ← BoxList − {B}
        else                                        ▷ Split B and continue checking.
            BoxList ← BoxList − {B}
            BoxList ← SplitBox(B) ∪ BoxList
        end if
    end while
    return True▷ If we have reached this point, R' was split into smaller boxes
that were found to be contained in index boxes.
end function
```

Since SPLITBOX eventually outputs boxes that are smaller than the input

box R', this procedure must terminate. In the worst case, this happens when BoxList contains boxes with side length at most the smallest side length of an exclusion or sample rectangle. If CheckRegion returns False, it does so while considering a box that does not intersect any sample or exclusion box contained in the search space. When the region being checked is R, the midpoint of this box is a candidate to serve as the return value of ReturnTestPoint. This candidate point is not within ϵ of any currently saved sample point, nor is it in a previously saved exclusion zone.

Note that the main **while** loop takes boxes from the end of BoxList, while smaller boxes returned by SplitBox are appended to the beginning of BoxList. This is an important point in the efficiency of this procedure, as it results performing a breadth-first search of the BoxList, rather than a depth-first search. In a depth-first search where BoxList has more than one element, the procedure does not start checking and splitting additional boxes until it confirms whether the very first box is covered by index rectangles or not. If this first box is covered, it could take several iterations of splitting to confirm this fact. Meanwhile, another one of the boxes in the original list might require very few iterations of splitting to confirm that it is not covered. The extra splitting and checking of the first box is wasted computation.

SplitBox's implementation can significantly affect the overall efficiency of CheckRegion. Let $B = \Pi_{i=1}^{N}[c_i, d_i]$ be the input provided to SplitBox. A straightforward method is to split B into much smaller boxes without using any external information. As an example of such a method, for a fixed dimension m in $\{1, \ldots, N\}$ we can split B into the two halves $B_1 = \Pi_{i=1}^{m-1}[c_i, d_i] \times [c_m, \frac{c_m+d_m}{2}] \times \Pi_{i=m+1}^{N}[c_i, d_i]$ and $B_2 = \Pi_{i=1}^{m-1}[c_i, d_i] \times [\frac{c_m+d_m}{2}, d_m] \times \Pi_{i=m+1}^{N}[c_i, d_i]$. We must take care that the dimension m rotates through $1, \ldots, N$ cyclically when repeatedly splitting boxes to fulfill the technical condition, but otherwise this procedure could serve as SplitBox in principle.

Although a single iteration of splitting has negligible computational expense with this approach, even the simple situation depicted in Figure 3.1 takes many splitting iterations to verify that the input box is not covered by index boxes. The uncovered rectangular region has also been divided into smaller boxes that would require nontrivial computational investment to identify and reunite into a single

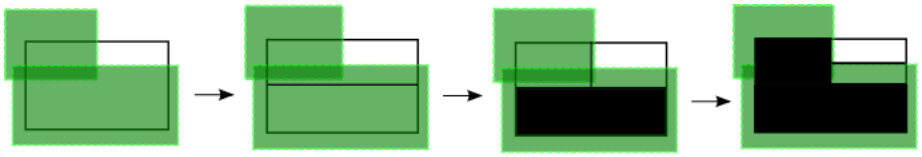


Figure 3.1: The state of BoxList as CheckRegion checks a simple example using a box halving variant of SplitBox. Each step represents the BoxList at the end of an iteration of the main **while** loop in CheckRegion. Green boxes are index boxes and are not in BoxList. Black boxes are boxes that have been removed from BoxList, and all other boxes are elements of BoxList.

box. Dividing the uncovered region makes it more difficult to select an optimal return value for ReturnTestPoint.

An adaptive procedure for SplitBox handles both these issues at the cost of some increase in computational time for splitting. Instead of splitting the box B into two halves arbitrarily, the procedure identifies the index box which has the largest intersection C with B according to the measure of C. In an edge case, the measure of C is 0 so that we can conclude that B is not covered by index boxes without needing any splitting. Otherwise, C is a box properly contained in B. The SplitBox procedure splits B into a list of smaller boxes that includes C while trying to keep the number of boxes low. Intuitively, this splitting procedure first cuts B into "strips" along each dimension, one of which contains C. It then repeats this process on the strip that contains C in the next dimension up. At the final iteration one of the strips obtained is C. See Figure 3.2.

Figure 3.2: Splitting a box B into a collection of smaller boxes $\mathcal{B} = \{B_1, B_2, \ldots, B_l\}$ such that a fixed subbox B' of B is contained in $\mathcal{B}$. The large box is B and the green box is B'. In the middle image, B has been cut into strips along the x dimension so that one of the strips contains B'. In the far right image, the strip containing B' has been further cut into strips along the y dimension, so that one of the strips is B'.

Using this variant of SplitBox in CheckRegion, the box C is added to the BoxList and subsequently removed since it is by definition covered by an index box. B is split into fewer, larger boxes, and so the boxes in BoxList and their midpoints are kept as good candidates to serve as the output for ReturnTestPoint. One last adjustment is made to CheckRegion to improve the output of ReturnTestPoint. We sort the boxes in BoxList by measure every time the **while** loop repeats, so that the largest boxes in BoxList are considered first. Combined with the breadth-first nature of the search, this ensures that ReturnTestPoint returns the midpoint of the largest box possible that does not intersect any index box, provided such a box exists.

3.3 Core procedure design

The main strategy for the SA utilizes the previously discussed components to obtain a set of index boxes which cover the overall bounds R as in Proposition 3.1. The following procedure makes repeated calls of the form MinPoint(y) while varying the test point $y \in R$, then adds the returned sample points and exclusion regions to the search space.

Algorithm 3.3 Core procedure

```
y ← random point in R
while CheckRegion(R) is False do
    for each real critical point p from MinPoint(y) do
        InsertSamplePoint(p)
    end for
    δ' ← minimum distance from MinPoint(y)
    InsertExclusionRegion(y, δ')
    y ← ReturnTestPoint()
end while
OutputSample()
```

This procedure will eventually terminate and output a sample fulfilling the density requirements, provided that ReturnTestPoint has been implemented as described in the previous section. A major practical issue is that MinPoint is computationally expensive relative to the number of times it needs to be called

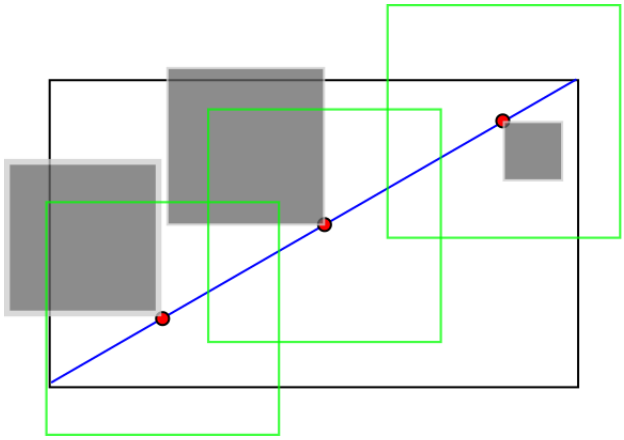


Figure 3.3: The state of an ongoing ϵ-sampling procedure for a variety $\mathcal{V}_{\mathbb{R}}(f)$ and overall bounds R. The blue line segment pictured is $\mathcal{V}_{\mathbb{R}}(f) \cap R$. Gray boxes are exclusion boxes, boxes with green outlines are sample boxes, and red points are sample points. The box with a black outline is R. Since the sample boxes cover $\mathcal{V}_{\mathbb{R}}(f) \cap R$, the sample points form an ϵ-sampling of $\mathcal{V}_{\mathbb{R}}(f) \cap R$. The sample and exclusion boxes together do not cover R, however.

in this version of the algorithm, particularly in higher dimensions. Another issue is that the algorithm could add far more sample points than necessary. Consider the situation in Figure 3.3. Though the sample points depicted in the Figure form an ϵ-sampling, the algorithm cannot confirm this fact because R is not covered by sample boxes and exclusion boxes. Given the algorithm is in the state pictured in Figure 3.3, the **while** loop in the procedure above will execute several more times before outputting a sample. Each iteration will add unnecessary sample points.

To address the computational waste of using MinPoint to calculate sample points, another step can be interjected before executing Algorithm 3.3 above. The step parallels Algorithm 3.3, but uses CritPoint instead of MinPoint to find sample points. Recall that if $y \in R$ is a test point and CritPoint(y) returns a critical point $x \in \mathcal{V}_{\mathbb{R}}(f)$, $y - x$ is orthogonal to the tangent hyperplane of $\mathcal{V}_{\mathbb{R}}(f)$ at x, provided that x is nonsingular. The vector $y - x$ can be used to project other vectors onto the hyperplane.

The above procedure chooses new test points by a type of local linear approximation to the underlying variety. Both the number of points to compute and the rescaling factor, s, are parameters for the procedure. In the implemented version

Algorithm 3.4 CritPoint sampling

$y \leftarrow$ random point in R
while Number of sample points computed is less than some threshold **do**
  $x \leftarrow \text{CritPoint}(y)$
  InsertSamplePoint(x)
  $u \leftarrow$ random unit vector starting at x with $u \not\perp y - x$
  $y \leftarrow$ projection of u onto hyperplane at x defined by normal $y - x$
  $y \leftarrow sy$ where s is a scaling factor
end while

$s = 1.5\epsilon$ is used. Some guidance on an upper bound to use for the number of sample points is given by the following theorem.

Theorem 3.5. *Let $R = \Pi_{i=1}^{N}[a_i, b_i]$. There is an ϵ-sampling $X_0 \subseteq \mathbb{R}^N$ of R which consists of at most $\Pi_{i=1}^{N} g_i$ points where $g_i = \lceil \frac{\sqrt{N}(b_i - a_i)}{2\epsilon} \rceil$.*

Proof. Let $c = \frac{2\epsilon}{\sqrt{N}}$. For each $i \in \{1, \ldots, N\}$ we define a set of points $G_i \subseteq \mathbb{R}$. If $c \geq b_i - a_i$, then $G_i = \{\frac{a_i + b_i}{2}\}$. Otherwise, $G_i = \{a_i + \frac{c}{2}, a_i + c(1 + \frac{1}{2}), a_i + c(2 + \frac{1}{2}), \ldots, a_i + c(\lfloor \frac{b_i - a_i}{c} - \frac{1}{2} \rfloor + \frac{1}{2}), b_i - \frac{c}{2}\}$. Since $\lfloor \frac{b_i - a_i}{c} - \frac{1}{2} \rfloor \leq \lfloor \frac{b_i - a_i}{c} \rfloor$ each set G_i contains at most g_i points. Now if $p \in [a_i, b_i]$ there is a point with distance at most $\frac{c}{2}$ from p in G_i. This implies that any point in R is at most distance $\sqrt{\frac{Nc^2}{4}} = \sqrt{N}\frac{c}{2} = \epsilon$ from some point in $G = \Pi_{i=1}^{N} G_i$. Therefore G is an ϵ-sample of R with the desired number of points. □

Let $M = \Pi_{i=1}^{N} g_i$ as given in Theorem 3.5. Since an ϵ-sampling of $\mathcal{V}_{\mathbb{R}}(f) \cap R$ can be obtained from an ϵ-sampling of R by removing any points in the sampling of R that are more than ϵ away from $\mathcal{V}_{\mathbb{R}}(f)$, M is an upper bound on the size of a minimal ϵ-sampling. We can choose some proportion of M as the number of sample points to collect using CritPoint sampling. In practice, we select this proportion as $1 - 9\%$ of M. The sampling in Theorem 3.5 corresponds to splitting R into a grid with pixels that fit into balls of at most radius ϵ, and selecting a proportion of M corresponds to making a guess for how many of those pixels intersect with $\mathcal{V}_{\mathbb{R}}(f)$. Performing CritPoint sampling followed by the core procedure both saves time on computing sample points, and potentially cuts down on the number of calls to MinPoint necessary to cover R with exclusion boxes. Without further

modification to the SA, adding this step only exacerbates the problem of adding extraneous sample points.

3.4 Heuristics for minimizing the sample

In the core procedure, MinPoint is used to fulfill two separate objectives simultaneously: computing sample points, and determining exclusion regions. The heuristics we introduce serve to decouple these two objectives from one another, while still taking some advantage of the fact that MinPoint produces both sample points and exclusion regions. Efficient splitting of boxes and selection of test points in the implementation of the search space already take a step towards this effort by being selective about which exclusion regions are calculated. Instead of concentrating on exclusion regions, this section's heuristics approach the problem by focusing on how sample points are added into the search space.

3.4.1 Skipping the addition of sample points

Suppose that a sample point $x \in \mathcal{V}_{\mathbb{R}}(f) \cap R$ with sample box R' is provided to the InsertSamplePoint function. If R' is covered by other sample and exclusion boxes already saved in the search space, then adding x to the search space with index box R' contributes nothing towards producing an ϵ-dense sampling. We adjust InsertSamplePoint so that it skips adding x and R' to the search space if CheckRegion(R') returns True.

The second heuristic is much less straightforward and relies on an assumption that in a minimal ϵ-sample the sample points are relatively far away from each other. Equivalently, the sample boxes corresponding to sample points do not overlap much. Based on this assumption, let C be the largest intersection of R' with a sample box already in the search space according to its measure. If the measure of C is greater than some threshold value $T > 0$, InsertSamplePoint refuses to save x as a sample point. The most optimistic choice for this threshold value is 0, as this corresponds to none of the sample boxes overlapping in the final sampling. There are counterexamples where the minimal ϵ-sampling of $\mathcal{V}_{\mathbb{R}}(f) \cap R$ must contain overlapping sample boxes, however. For instance, suppose that

$\mathcal{V}_{\mathbb{R}}(f) \cap R$ is a line segment of length $\alpha + \beta$ with $0 < \beta < \alpha$. Any α sample of this line segment contains at least two points and the two points' sample boxes overlap. This problem persists when choosing any T less than the measure of a sample box. If β in the line segment example is taken small enough, the measure shared by the overlapping sample boxes can be taken as close to the measure of a sample box as desired.

If R' is the sample box which InsertSamplePoint has constructed before adding a sample point to the search space, let M_C be the measure of C, where C is the largest intersection by measure of R' with a sample box already in the search space. Also let M_ϵ be the measure of a sample box. Given the threshold T is chosen to be M_ϵ or greater, no sample points will be skipped over. Choosing such a value for T therefore does not change the SA's behavior, which is undesirable. To consult these issues of setting T too large or too small, we make T a dynamic variable rather than a static one. We initialize T at a value less than M_ϵ. If too many of sample points which the algorithm attempts to add to the search space fail the condition $M_C \leq T$, then T is increased by some small amount. Eventually, either the SA outputs a sample, or T reaches the value M_ϵ.

To determine whether too many sample points are skipped, we can choose integer parameters P and S with $0 < P \leq S$. If P of the last S sample points provided to InsertSamplePoint were skipped, then T is increased. Selecting the value of P is a trade off. A higher value potentially wastes more computation by skipping many useful sample points before adjusting T, and a lower value could result in inserting too many sample points. Choosing higher values for S reduces the chance that a small number of calls to MinPoint at test points which are close to one another causes an adjustment to T, but it might very well be necessary to increase T to add a required sample point. In the implemented version of the algorithm $S = 50$, $40 \leq P \leq 50$, T starts at value $\frac{M_\epsilon}{2.5^N}$, and T increments by subtracting 0.2 from the denominator at each step.

3.4.2 Adjusting the core procedure

As a variant on the skipping strategy of the previous section, we can also add another step to the SA between CritPoint sampling and the core procedure.

The new step executes the same way as the core procedure, but does not attempt to add most of the sample points returned by MinPoint to the search space. Instead, it only inserts an exclusion region in each repetition of its main **while** loop. Only if the computed exclusion region has measure smaller than a sample box's does the new step also attempt to insert the sample point x returned by MinPoint(y) which minimizes the distance $\|y - x\|$ from the test point to $\mathcal{V}_{\mathbb{R}}(f)$.

Running the algorithm in this mode skips adding points as with the heuristic detailed in the previous section. The intent is to decrease the amount of space in R that needs to be covered by exclusion regions before the algorithm starts inserting sample points. Deciding how long to run the algorithm in this mode generates another parameter to select. We run the SA using this new step until the maximum measure of a box in BoxList is less than $T' > 0$ for a second threshold value T'. When all the boxes in BoxList are below this size they should be more readily covered by sample boxes. In practice, we choose values for T' ranging from $\frac{M_\epsilon}{25\times 10^4}$ to M_ϵ.

3.5 Extension to real semialgebraic sets

A few modifications are necessary to extend the SA to real semialgebraic sets. In the first case, we need to extend the current version of the SA from requiring pure d-dimensional varieties as input to accepting arbitrary varieties. Theorem 2.12 proves that the homotopy utilized by MinPoint provides solutions to the minimum distance problem on any pure d-dimensional subvariety $V \subseteq \mathcal{V}_{\mathbb{R}}(f)$, provided that f is a polynomial system with real coefficients from $\mathbb{C}^N$ to $\mathbb{C}^{N-d}$. For a variety that is defined by a system of polynomials with real coefficients $g : \mathbb{C}^N \to \mathbb{C}^n$ with $n \geq N$ that is not pure dimensional, we can obtain a new system $g' : \mathbb{C}^N \to \mathbb{C}^{N-d}$ using the randomization procedure described in Section 2.1.3. The variety obtained used this randomization procedure will have V as an irreducible component of $\mathcal{V}(g')$ for generic randomization parameters, though $\mathcal{V}(g')$ may also contain irreducible components which are not irreducible components of $\mathcal{V}(g)$. Assuming that $\mathcal{V}(g)$ has a d-dimensional component, the system g' can be provided to the algorithm to obtain an ϵ-sampling of the pure d-dimensional component of $\mathcal{V}_{\mathbb{R}}(g)$. Repeating this for every dimension d such that $\mathcal{V}(g)$ has a

d-dimensional component and combining the samples gives an ϵ-sample of $\mathcal{V}_{\mathbb{R}}(g)$. We must perform additional computations to filter out points returned by the SA that are elements of $\mathcal{V}_{\mathbb{R}}(g')$, but not elements of $\mathcal{V}_{\mathbb{R}}(g)$.

Suppose we have polynomials with real coefficients $h_1, \ldots, h_n, p_1, \ldots, p_m : \mathbb{C}^N \to \mathbb{C}$ that define the real semialgebraic set S of points $x \in \mathbb{R}^N$ where all the h_i have $h_i(x) = 0$ and all the p_i have $p_i(x) \geq 0$. Then if $h : \mathbb{C}^N \to \mathbb{C}^n$ is the system corresponding to the h_i, we can apply the SA to $\mathcal{V}_{\mathbb{R}}(h) \cap R$, but only add points to the sample which fulfill the inequality constraints. Since for any $y \in R$ we have

$$\min_{x \in \mathcal{V}_{\mathbb{R}}(h)} \|x - y\|^2 \leq \min_{x \in S} \|x - y\|^2,$$

it follows that exclusion regions for $\mathcal{V}_{\mathbb{R}}(h)$ are also exclusion regions for S. Although there may be extra computation involved to account for the inequalities, the algorithm still outputs a valid ϵ-sample of $S \cap R$.

3.6 Implementation and parallelization

3.6.1 Implementation

The SA described in the previous sections has been implemented as a Python module utilizing C and C++ bindings for major computational components. Source code for the algorithm and examples is available at `http://github.com/P-Edwards/sampling_varieties`. It employs an R^* tree spatial database for the search space.

One of the major practical challenges in implementing the SA is integrating external software packages with other program components, and particularly with using the package `Bertini`. The library used to provide the R^* tree has very specific set up requirements. We use `Bertini` to perform homotopy continuation computations. It provides robust and parallelizable methods for doing this, but is controlled exclusively through terminal commands and reading and writing to text files rather than a programmatic interface. As noted in Chapter 2, Bertini has several internal parameters which rely on detailed knowledge of the homotopy continuation process to choose properly. Every call to MinPoint or CritPoint requires the construction of an input file specific to that call's test point. `Bertini` does not expose any internal variables to external program components as a result

of the input and output design, which presents some unique challenges. The configuration files for CritPoint require a specially tailored input template for each example because of this. This is also the case for MinPoint, but its template is not as complicated as CritPoint's.

As with many computational problems having Euclidean space as a setting, sampling real algebraic varieties suffers from the so-called "curse of dimensionality". With every additional dimension, required computational resources increase exponentially. As an example of this, a single box in N dimensional Euclidean space takes $128 \cdot N$ bits to store. For $N = 12$, this means that approximately 5×10^6 boxes require one gigabyte of RAM to store. To put this in perspective, if the region $[0, 20]^{12}$ is split into equally sized hypercubes with diagonal length 4.8, the number of boxes is around 5×10^6.

The specific strategies provided in Section 3.2.2 for the SplitBox and CheckRegion functions demand a very careful design which allows samplings to be computed despite these resource constraints. Both functions avoid splitting larger boxes into many small boxes when at all possible. This has the added advantage of producing good values for ReturnTestPoint which help avoid unnecessary calls to MinPoint. An unrefined procedure for ReturnTestPoint can easily result both in wasting computation time calling MinPoint, and also in adding many very small exclusion boxes to the search space which slow down subsequent computation. Earlier versions of the implementation lacking careful strategies produced exactly these results, and could not compute samples at all.

Dimensionality considerations also affect the tuning of parameters for the various heuristics. In higher dimensions the measure of different sized boxes relative to one another varies more as a function of side length than in lower dimensions. In dimension 12, a hypercube with sides of length .95 has measure .54, even though its side length only varies .05 from 1. In contrast, a square in dimension 2 with side length of .95 has measure .90. Heuristics parameters can be tuned on an example-by-example basis by first computing a very coarse sampling, then using it to inform the tuning. Using the maximum length of a side of a box as its measure rather than Lebesgue measure is also supported by the implementation. In practice, the Lebesgue measure performs better.

Several other optimizations serve to reduce computation times. The BoxList in CheckRegion is saved between calls to CheckRegion when the original bounds R are being checked, which saves on the computational costs of splitting R repeatedly. Sorting the BoxList in CheckRegion quickly becomes computationally expensive for longer lists, so the implemented version only does this sorting once every 30 times an exclusion or sample box is added to the search space. Instead of calculating the measure of each box in BoxList every time the list is sorted, the measure of each box is paired with the box and stored in the BoxList.

3.6.2 Parallelization

Computational speedups on modern hardware rely on executing independent sections of a computationally expensive problem in parallel on multiple processing elements. As noted previously, the most expensive segment of the algorithm consists of calls to MinPoint for determining exclusion regions and sample points. Homotopy continuation can be parallelized, with the different paths in the path tracking roughly corresponding to independent problems into which the continuation can be split. Running MinPoint in parallel using `Bertini` realizes significant speed gains. There are diminishing returns on the number of processing elements provided to a single instance of homotopy continuation, however. Homotopy continuation parallelization is the primary parallelization used by the current implementation.

The algorithm can be further parallelized by running multiple calls to MinPoint simultaneously, allocating some number of processors to each. A master process maintains the search space and calculates new test points. Homotopy continuation for CritPoint suffers more quickly from diminishing speed returns on the number of processors allocated to an individual instance of continuation since there is only one path to track in this case. We can split the number of sample points we wish to calculate with CritPoint sampling evenly between several parallel executions of the sampling process.

It is possible to split the region R into smaller subregions and execute completely parallel instances of the entire sampling algorithm on the subregions. Combining the samples output by the SA on the subregions gives an ϵ-sampling of

$\mathcal{V}_{\mathbb{R}}(f) \cap R$. This strategy suffers drawbacks, however, in that separate instances of the algorithm maintain their own versions of the search space and point skipping heuristics. Parallelizing in this way primarily saves on computation time devoted to making calls to a somewhat larger spatial database underlying the search space unless diminishing returns from allocating additional processors to homotopy continuation are high. The current implementation does support subregion level parallelization. In practice, the gains usually do not outweigh the drawbacks, particularly the potential to increase the size of the sampling.

Chapter 4

Results II: Examples

This chapter presents the results of applying the sampling algorithm described in Chapter 3 to several polynomial systems. After obtaining as dense a sample as possible with a sufficiently small number of points, TDA computations were performed to determine persistence diagrams. Where possible, we interpret the persistence diagrams using Corollary 2.34 to infer the existence of homology features in the underlying varieties from the persistence diagrams. We assume that the condition on the homological feature size of the underlying variety given in that corollary is true for all examples. Source code for the examples is available along with the source code for the algorithm implementation at `github.com/P-Edwards/sampling_varieties`.

The software package `DIPHA` is employed for persistent homology calculations. Persistent homology is calculated only up to dimension 3 due to computational constraints. This means that only 3 dimensional holes and lower can be identified. All computations performed for timing were executed on a shared memory system. The machine runs Ubuntu version 16.04.1 as the operating system. It utilizes 32 computing threads at 3.30GHz split evenly among 16 processing elements contained in 2 physical CPU's, along with 252GB of RAM.

4.1 Example: Alpha curve

We start with a simple example to test the algorithm in a situation where the variety and sample points are as easily visualized as possible. Let $f \in \mathbb{R}[x_1, x_2]$ be

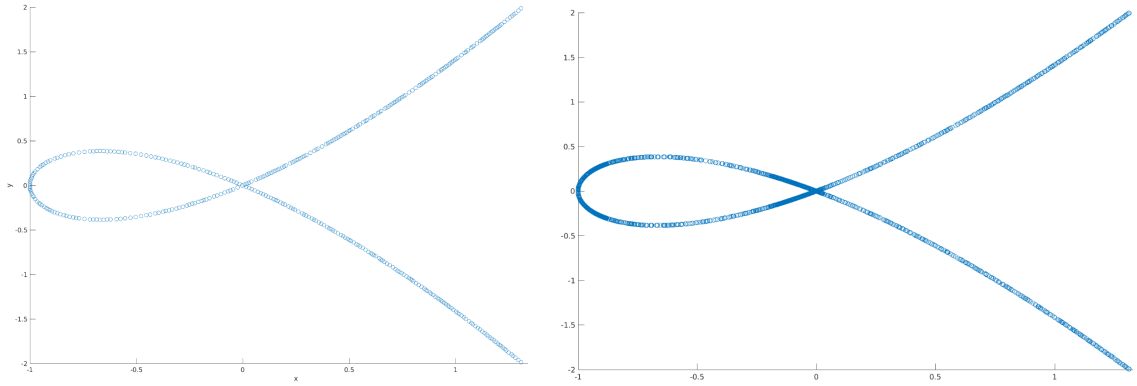


Figure 4.1: Two .02-samplings for $\mathcal{V}_{\mathbb{R}}(f)$. The sampling on the left was collected with heuristics, the sampling on the right without.

the polynomial $x_2^2 - x_1^2(x_1 + 1)$, which is the "alpha curve" pictured in Figure 2.1. Note that $\mathcal{V}(f)$ is a pure 1-dimensional variety, since f is irreducible over $\mathbb{C}^2$. The parameter homotopy from Theorem 2.12 is given by (where $(y_1, y_2)^T = y \in \mathbb{R}^2$ is a generic test point, $(a_1, a_2)^T$ is any point in $\mathbb{C}^2$, z is a generic point in $\mathbb{R}$, $\gamma, \alpha_1, \alpha_2$ are generic points in $\mathbb{C}$):

$$H(a_1, a_2, \lambda_0, \lambda_1, t) = \begin{pmatrix} f(a_1, a_2) - t\gamma z \\ \lambda_0(a_1 - y_1) + \lambda_1(3a_1^2 + 2a_1) \\ \lambda_0(a_2 - y_2) + \lambda_1(2a_2) \\ \alpha_0\lambda_0 + \alpha_1\lambda_1 - 1 \end{pmatrix}.$$

The SA was used to find a .02 sampling of $\mathcal{V}_{\mathbb{R}}(f) \cap R$ where $R = [-2, 2] \times [-2, 2]$. It returned a sample consisting of 387 points in $[-2, 2] \times [-2, 2]$. The theoretical maximum sample size given by Theorem 3.5 for density .02 is 20164, and running the sampling algorithm without heuristics returned a sampling of size 3962. 3962 points is too large a number to use for computing TDA due to resource constraints, even when limiting calculations to 2 dimensional homology. Sampling without heuristics took 2 minutes and 55 seconds of real time, while sampling with heuristics took 5 minutes and 52 seconds. Figure 4.1 displays the two sets of sample points, and Figure 4.2 displays the persistence diagram derived from the smaller sampling.

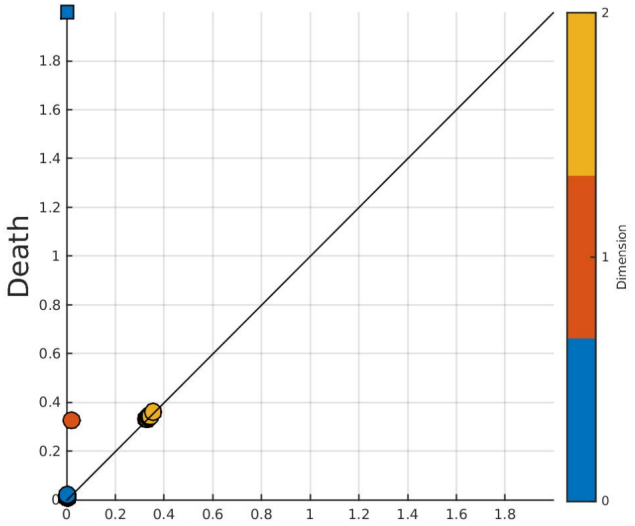


Figure 4.2: Persistence diagram derived from a .02 sampling of $\mathcal{V}_{\mathbb{R}}(f) \cap R$.

The number of points in the persistence diagram that are above and to the left of $(.017, .06)$ in a given dimension $k \geq 0$ is a lower bound on the rank of the homology group $H_k(\mathcal{V}_{\mathbb{R}}(f) \cap R)$ in that dimension. As expected, we find that $H_0(\mathcal{V}_{\mathbb{R}}(f) \cap R)$ and $H_1(\mathcal{V}_{\mathbb{R}}(f) \cap R)$ have at least rank 1. The feature in dimension 1 corresponds to the loop in the alpha curve, and the feature in dimension 0 corresponds to the single connected component of $\mathcal{V}_{\mathbb{R}}(f) \cap R$.

4.2 Example: Quartic varieties

Our next set of examples moves up a dimension and applies the combined sampling/TDA process to 16 different quartic equations in 3 variables. It can be checked using NAG methods that each of the following polynomial equations defines a pure 2-dimensional algebraic variety. Letting $N = 3$ and $d = 2$, we see that each polynomial is a polynomial system from $\mathbb{C}^N$ to $\mathbb{C}^{N-d}$. The algorithm's assumptions on the number of equations therefore hold without further modification of the polynomials.

Persistence diagrams for these examples are available in Appendix A. We consider a few of the varieties in more detail. Let $V_4 = \mathcal{V}_{\mathbb{R}}(7x^4 + 4y^4 + 8z^4 - 5x^3 + 8x^2y + 5x^2 - 7xy^2 - 5xy + 4x + 4y^3 - y^2 - 3y) \cap [-2, 2] \times [-2, 2] \times [-2, 2]$, which corresponds to the fourth row in the table. It is the only example which definitely has

Variety	Defining equation
V_1	$5x^4+6y^4+6z^4-7+2x^3+2x^2y+x^2+3xy^2-5xy-8x-8y^3-y^2-3y$
V_2	$144x^4+144y^4-225(x^2+y^2)z^2+350x^2y^2+81z^4+6-7x^3+3x^2y-4x^2-6xy^2-xy+6x+7y^3-3y^2-8y$
V_3	$7x^4+10y^4+6z^4+2-7x^3+3x^2y-5x^2-3xy^2-4xy+8x-y^3-4y^2-7y$
V_4	$7x^4+4y^4+8z^4-5x^3+8x^2y+5x^2-7xy^2-5xy+4x+4y^3-y^2-3y$
V_5	$4x^4+4y^4+3z^4+2-5x^3+6x^2y-7x^2+4xy^2-3x-8y^2-4y$
V_6	$x^4+8y^4+8z^4-6+8x^3-8x^2-7xy^2-4xy-2x-3y^3+5y$
V_7	$10x^4+y^4+8z^4-5-8x^3+x^2y-8x^2+5xy^2-2xy-2x-6y^3+4y^2-3y$
V_8	$2x^4+9y^4+10z^4+4+4x^3+3x^2y-4x^2-6xy+7x+7y^3+7y^2-7y$
V_9	$4x^4+7y^4+3z^4-3-8x^3+2x^2y-4x^2-8xy^2-5xy+8x-6y^3+8y^2+4y$
V_{10}	$144x^4+144y^4-225(x^2+y^2)z^2+350x^2y^2+81z^4+x^3+7x^2y+3x^2+3xy^2-4x-5y^3+5y^2+5y$
V_{11}	$144x^4+144y^4-225(x^2+y^2)z^2+350x^2y^2+81z^4+3-4x^3-8x^2y+2x^2-2xy^2-2xy-5x+5y^3+5y^2+y$
V_{12}	$3x+7y-z^2(225x^2+225y^2)+350x^2y^2+5xy+xy^2+6x^2y-x^2+5x^3+144x^4+4y^2+144y^4+81z^4-8$
V_{13}	$144x^4+144y^4-225(x^2+y^2)z^2+350x^2y^2+81z^4-1+x^3-x^2y-5x^2+8xy^2-7xy+5x-y^3-6y^2+6y$
V_{14}	$144x^4+144y^4-225(x^2+y^2)z^2+350x^2y^2+81z^4+7-x^3+6x^2y+2x^2+6xy^2-4xy-7x+y^3-5y^2+y$
V_{15}	$144x^4+144y^4-225(x^2+y^2)z^2+350x^2y^2+81z^4+7x^3-3x^2y+4x^2+6xy^2-3xy-2x+y^3-2y^2-2y$
V_{16}	$144x^4+144y^4-225(x^2+y^2)z^2+350x^2y^2+81z^4-7-4x^3+4x^2y-2x^2+2xy^2-6xy+3x+8y^3-4y^2+y$

Variety	Region	Density	# of points	C_ℓ	C_u	β_0	β_1	β_2
V_1	$[-10,10]^3$	0.38	416	0.310	1.14	1	0	0
V_2	$[-5,5]^3$	0.98	422	0.800	2.94	1	0	0
V_3	$[-10,10]^3$	0.3	276	0.244	0.9	1	0	0
V_4	$[-2,2]^3$	0.15	472	0.122	0.45	2	0	0
V_5	$[-10,10]^3$	0.5	426	0.408	1.5	1	0	0
V_6	$[-10,10]^3$	0.9	409	0.734	2.7	1	0	0
V_7	$[-10,10]^3$	0.6	437	0.489	1.8	1	0	0
V_8	$[-10,10]^3$	0.3	467	0.244	0.9	1	1	0
V_9	$[-3,3]^3$	0.36	409	0.293	1.08	1	0	0
V_{10}	$[-5,5]^3$	1	503	0.8164	3	1	0	0
V_{10}	$[-5,5]^3$	1	516	0.816	3	1	0	0
V_{11}	$[-5,5]^3$	1	449	0.816	3	1	0	0
V_{12}	$[-5,5]^3$	1	571	0.816	3	1	0	0
V_{13}	$[-5,5]^3$	1	538	0.816	3	1	0	0
V_{14}	$[-5,5]^3$	1	518	0.816	3	1	0	0
V_{15}	$[-5,5]^3$	0.97	510	0.97	2.91	1	0	0

Table 4.1: Sampling and TDA results for 16 varieties defined by quartic equations in the variables x, y, and z. The C_ℓ and C_u columns record the point $(C_\ell, C_u) \in \bar{\mathbb{R}}^2$ such that underlying homology features of the variety appear above and to the left of (C_ℓ, C_u) in the persistence diagram for the sampling. The numbers β_0,β_1, and β_2 record the number of points above and to the left of (a, b) in the persistence diagram for dimensions 0, 1, and 2 respectively.

at least 2 connected components ($\beta_0 = 2$ in the table). Its .14-sampling is pictured in Figure 4.3. Notice that no sample point is within distance .14 of the boundary of the region. Combining this with the fact that no sample points were found to be outside of $[-2, 2] \times [-2, 2] \times [-2, 2]$ during the sampling, we can conclude that Figure 4.3 displays a sampling of all of V_4.

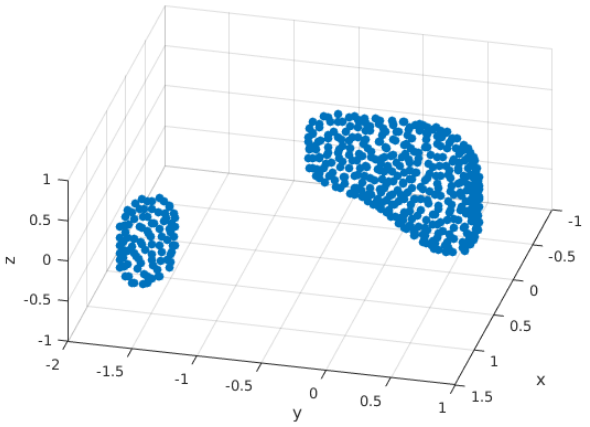
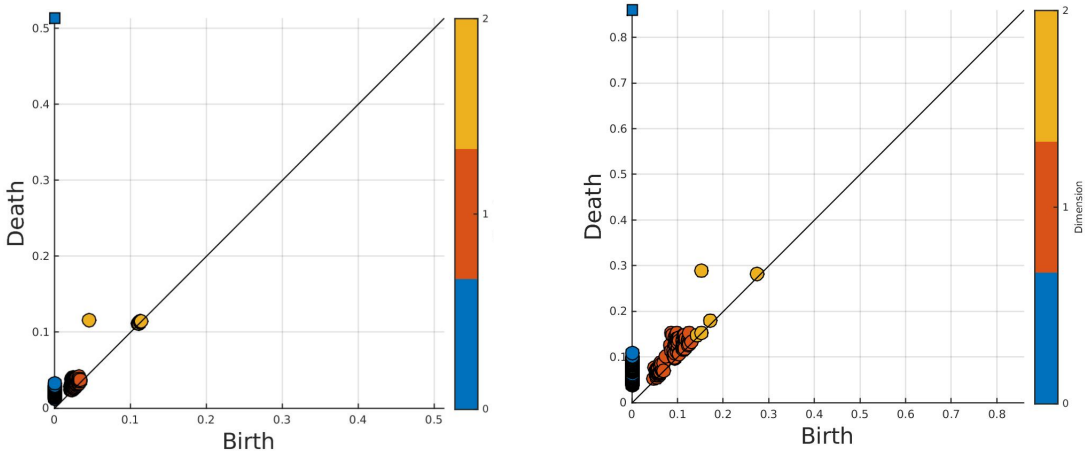


Figure 4.3: A .14-sampling of $\mathcal{V}_{\mathbb{R}}(7x^4 + 4y^4 + 8z^4 - 5x^3 + 8x^2y + 5x^2 - 7xy^2 - 5xy + 4x + 4y^3 - y^2 - 3y)$.

The points visually appear to be sampled from a space that is (up to homotopy equivalence) the union of two spheres. No 3D voids appear as confirmed homology features in the persistence diagram for V_4, however. To investigate this situation further, we apply sampling and TDA to $S_1 = V_4 \cap [0.5, 2] \times [-2, -1] \times [-0.7, 1]$ and $S_2 = V_4 \cap [-1, 0.5] \times [-0.8, 1] \times [-0.7, 1]$ independently. The restrictions allow us to obtain samplings at smaller densities for the individual pieces while still calculating point samples with few enough points to do TDA. The persistence diagrams derived from samplings of S_1 and S_2 are depicted in Figure 4.4.



Persistence diagram for S_1 Persistence diagram for S_2

Figure 4.4: Persistence diagrams for S_1 and S_2.

As expected, the persistence diagrams for both S_1 and S_2 contain a point in dimension 2, where the point is relatively far away from the diagonal. The different scales for S_1 and S_2 correspond to a higher sampling density for S_1, which was sampled at density .06 compared to density .1 for S_2. These points potentially correspond to 3D voids in the underlying spaces S_1 and S_2. The points do not meet the criterion from Corollary 2.34, and so we cannot be sure with theoretical certainty that $H_2(S_1)$ and $H_2(S_2)$ both have at least rank 1. Despite this, we have good evidence that V_4 is topologically the union of two spheres since we also know for certain that V_4 has two connected components.

From Table 4.1 we see that TDA results provide theoretical verification of homology features for very few of the varieties. Despite this, several of the persistence diagrams in Appendix A have one or two points which are significantly farther from the diagonal than the others in their persistence diagrams. This situation displays the difficulty of obtaining dense enough samples to verify features formally, and also that TDA can provide useful information even when we cannot prove the Homology Inference Theorem applies.

We can contrast V_4 with $V_9 = \mathcal{V}_{\mathbb{R}}(2x^4 + 9y^4 + 10z^4 + 4 + 4x^3 + 3x^2y - 4x^2 - 6xy + 7x + 7y^3 + 7y^2 - 7y) \cap [-3, 3]^3$, which corresponds to row 9 in Table 4.1. Its

persistence diagram and .32 sampling are pictured in Figure 4.5. Similarly to V_4, the entire variety $\mathcal{V}_{\mathbb{R}}(2x^4+9y^4+10z^4+4+4x^3+3x^2y-4x^2-6xy+7x+7y^3+7y^2-7y)$ is seen to be contained in $[-3,3]^3$. We might initially guess from the persistence diagram for V_9 that the variety consists of two spheres up to homotopy. Unlike V_4, the persistence diagram does not indicate that V_9 contains more than one connected component. Upon visual inspection of the sampling in Figure 4.5, V_9 appears to be a sphere up to homotopy equivalence, though two voids are recognizable. V_9's persistence diagram captures some of this geometric information, even though the homology features in the persistence diagram probably do not correspond with homology features in the space V_9.

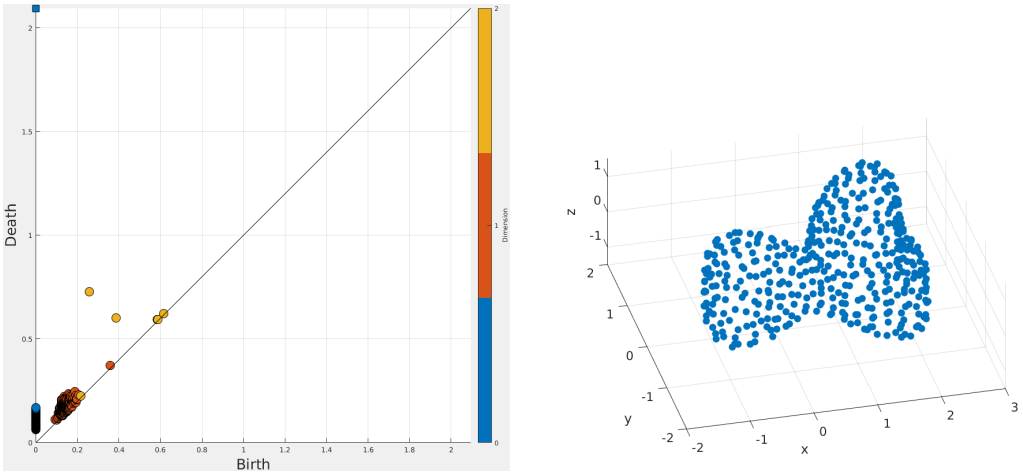


Figure 4.5: Persistence diagram and .36 sampling for V_9.

The process we applied to V_4 can be generalized to higher dimensions. For a variety in more than 3 dimensions for which we have confirmed the existence of a number of connected components, we can apply clustering methods to isolate points in each of the connected components. Clustering methods receive as input a set of points in Euclidean space, and output a partition of the set of points into different clusters based on the points' proximity. This is an alternative to determining connected components visually, as we did with V_4.

A clear pattern in Table 4.1 is that samples for the polynomials containing (x^2+y^2) terms have a higher number of points in a smaller sample region and at higher densities than the other polynomials. Figure 4.6 shows a sampling of one

of the varieties in this class. The variety does not appear to be bounded, whereas the varieties without $(x^2 + y^2)$ terms are bounded.

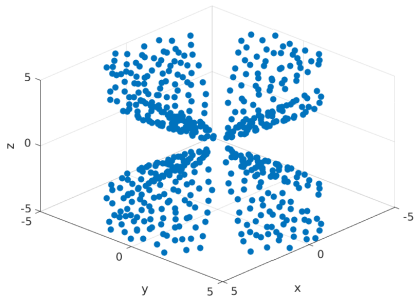


Figure 4.6: A .88 sampling of $V_2 = \mathcal{V}_{\mathbb{R}}(144x^4+144y^4-225(x^2+y^2)z^2+350x^2y^2+81z^4+6-7x^3+3x^2y-4x^2-6xy^2-xy+6x+7y^3-3y^2-8y) \cap [-5,5]^3$.

The larger sample sizes of these varieties at much higher density values emphasizes the difficulty of obtaining provably dense samplings containing a reasonable number of points. They also help illustrate the advantage of tuning the various heuristic parameters. The parameters for $V_{16} = \mathcal{V}_{\mathbb{R}}(144x^4+144y^4-225(x^2+y^2)z^2+350x^2y^2+81z^4-7-4x^3+4x^2y-2x^2+2xy^2-6xy+3x+8y^3-4y^2+y) \cap [-5,5]^3$ were set to skip points more aggressively than the other unbounded varieties. With these more aggressive parameters the sample sizes were decreased significantly. Without heuristics at all, the sampling algorithm returned a .97-sampling containing 13155 points for V_{16}.

4.3 Example: Configuration space of cyclooctane

Cyclooctane is an organic molecule with molecular formula C_8H_{16}. The 8 carbons in the molecule are bonded together into a ring, with 2 hydrogens bonded to each carbon atom. The C_8H_{16} molecule can take on different configurations which describe the positions of the individual atoms in relation to one another. Each configuration has an associated potential energy. Given a configuration of the carbon atoms, a full configuration for the molecule can be obtained by placing each carbon's two hydrogen atoms in a position minimizing potential energy. We

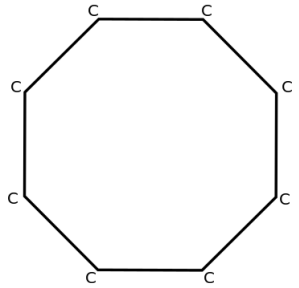


Figure 4.7: The molecular diagram of cyclooctane.

take a configuration to be represented by a set of squared distances between the 8 carbon atoms.

Assume that the following two constraints hold for the carbon atoms in cyclooctane. Firstly, the bond length between any two carbon atoms which are adjacent on the ring is the same for all the atoms. Secondly, the bond angle formed by any three consecutive atoms on the ring is the same. Since the bond length between adjacent carbons is fixed, fixing the bond angle is equivalent to fixing the distance between any two carbon atoms which have a single carbon atom between them on the ring. The assumption that these constraints lead to reasonable models of the molecule is called the "rigid geometry hypothesis", and it is justified by physical considerations as explained in [21]. Subject to these constraints, there are $\binom{8}{2} - 8 \cdot 2 = 12$ free inter-atom distances between carbon atoms in cyclooctane.

Let the carbon atoms in a molecule of cyclooctane be labelled cyclically by $\{1, \ldots 8\}$ and d_{ij} be the (squared) distance between atoms i and j in the molecule. A configuration of cyclooctane can be represented by a vector

$$\begin{pmatrix} d_{14}, & d_{15}, & d_{16}, & d_{25}, & d_{26}, & d_{27}, & d_{36}, & d_{37}, & d_{38}, & d_{47}, & d_{48}, & d_{58} \end{pmatrix}^T$$

in $\mathbb{R}^{12}$ designating the 12 unknown squared distances in the molecule. The *configuration space* S of cyclooctane is the set of vectors in $\mathbb{R}^{12}$ which designate a configuration of C_8H_{16} that is consistent with the rigid geometry hypothesis.

Understanding the topology and geometry of cyclooctane's configuration space can yield valuable information about the molecule. Paths in the configuration space correspond to a sequence of transitions from one configuration to another. Viewing potential energy as a function $P : S \to \mathbb{R}$, local minima of P potentially

correspond to more stable configurations of the molecule. The graph of P is the *potential energy landscape* of cyclooctane. Paths in the potential energy landscape which do not cross high energy regions represent configuration transitions that are energetically possible or more likely to occur. Topological information can lend insight leading to reducing the dimensionality S, which entails finding a description of S which uses fewer than 12 free variables. Reducing the dimensionality of S can then allow significant speed ups to calculations involving potential energy computations.

In [21] the authors perform an analysis of cyclooctane's configuration space using the local dimensionality reduction technique `ISOMAP` [27] and other tools. They determine that the configuration space requires 5 dimensions to represent, and that the space geometrically consists of a Klein bottle and a sphere which intersect along two circles. A significant step in their analysis consists of performing point set triangulation on a set of sample points from the configuration space, which shares similarities with the Čech complex and Rips complex constructions in Definition 2.21. Point set triangulation takes in a set of points as input and attempts to output a simplicial complex directly that closely matches the underlying space. This is in contrast to persistent homology, which obtains homology information from a sequence of complexes.

The configuration space of cyclooctane can be expressed as a semialgebraic set. We summarize a model of the cyclooctane molecule which yields a formulation of this semialgebraic set from [25]. We then detail attempts to perform sampling and TDA on the configuration space.

4.3.1 Distance model

For convenience, let the bond length between adjacent carbon atoms in cyclooctane be 1, and let the bond angle between any 3 consecutive carbon atoms be 115°. Rescaling the bond length this way does not affect the topology of the configuration space. Suppose we have a point $c \in \mathbb{R}^{12}$, and that c_{kl} is the squared distance between atoms k and l in the configuration of C_8H_{16} derived from c for any $k, l \in \{1, \ldots, 8\}$. The configuration c is contained in S if and only there exists a set of

8 points in $\mathbb{R}^3$ where the 8 points have the same squared distances between them derived from c.

If $\{a_1, \ldots, a_k\}$ is a subset of $\{1, \ldots, 8\}$, let $\delta_{ij} = c_{a_i a_j}$ for any $i, j \in \{1, \ldots, k\}$. The *Cayley-Menger determinant* of $\{a_1, \ldots, a_k\}$ is:

$$CM(a_1, \ldots, a_k) = \begin{vmatrix} 0 & \delta_{12} & \delta_{13} & \ldots & \delta_{1k} & 1 \\ \delta_{21} & 0 & \delta_{23} & \ldots & \delta_{2k} & 1 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ \delta_{k1} & \delta_{k2} & \delta_{k3} & \ldots & 0 & 1 \\ 1 & 1 & 1 & \ldots & 1 & 0 \end{vmatrix}$$

Notice that a Cayley-Menger determinant produces a polynomial in the squared distances between carbon molecules. Evaluating this determinant essentially returns a multiple of the signed Lebesgue measure of a $k-1$ simplex in some finite dimensional Euclidean space. The simplex has as vertices points with squared distance between them given by c. We have that c is a member of S if and only if there exist 4 elements $\{p_1, \ldots, p_4\}$ $\{1, \ldots, 8\}$ where:

$$\begin{aligned} CM(p_1, \ldots, p_4) &> 0 \\ CM(p_1, \ldots, p_4, i) &= 0 \\ CM(p_1, \ldots, p_4, j) &= 0 \\ CM(p_1, \ldots, p_4, i, j) &= 0 \end{aligned}$$

for all $i < j$ elements of $\{1, \ldots, 8\} - \{p_1, \ldots, p_4\}$.

If there was a set of 4 carbon atoms in cyclooctane that fulfilled the first inequality no matter the configuration in question, we could use that set to derive a description of S as a semialgebraic set. No 4 carbon atoms exist with this property in cyclooctane. Instead, it is shown in [25] that, if m is the label of any carbon atom in the molecule, at least one of the three sets of 4 atoms $\{m, m+1, m+2, m+3\}$, $\{m+1, m+2, m+3, m+4\}$, and $\{m+2, m+3, m+4, m+5\}$ (labels greater than 8 wrap around cyclically) will satisfy the first inequality for any configuration. This happens even though all three sets of atoms fail to satisfy the first condition for every configuration. Using this fact, we can find polynomial conditions which exactly describe every valid configuration of C_8H_{16}. This is

done by putting together the above listed conditions for $\{1,2,3,4\}$, $\{2,3,4,5\}$, $\{3,4,5,6\}$. We obtain a semialgebraic set with 3 inequalities and 22 equalities in 12 variables.

4.3.2 Model calculation and sampling configuration

Sampling and performing TDA for the set S requires significantly more effort than the previous examples. The polynomial system considered is larger, it is semialgebraic rather than algebraic, and the equations themselves are unmanageable by hand. It is also computationally expensive independent of the sampling algorithm's heuristics. In [25] the authors report using 72 hours of real time on a supercomputing platform while employing a sampling method which does not minimize sample points. Let $f : \mathbb{C}^{12} \to \mathbb{C}^{22}$ be the polynomial system consisting of the equalities for the cyclooctane distance model.

Recall from Chapter 3 that in order to apply the sampling algorithm we need to know the dimensions of the irreducible components of $\mathcal{V}(f)$. To obtain this information we can compute the *irreducible numerical decomposition*, which `Bertini` supports. Calculating the irreducible numerical decomposition is a feature of `Bertini` which requires independent setup from homotopy continuation. The results of this computation show that $\mathcal{V}(f)$ is a pure 2-dimensional variety.

To apply the homotopies from Theorems 2.12 and 2.13 we need to convert f into a system $g : \mathbb{C}^{12} \to \mathbb{C}^{10}$ since $\mathcal{V}(f)$ is pure 2-dimensional. As noted in Chapter 2, this requires choosing $22 \cdot 10 = 220$ random real numbers, and then using them to obtain the randomized system g which contains 10 polynomials consisting of linear combinations of the original 22 functions. As a technical note, the coefficients of the linear combinations must be real so that the reduced polynomial system obtained still has real coefficients. Randomization also introduces additional computational overhead while sampling, as we need to check whether the sample points we obtain are members of $\mathcal{V}_{\mathbb{R}}(f)$. The most straightforward way to do this is to evaluate every sample point using the original system. We must also evaluate potential sample points using the 3 inequality polynomials, and filter out those points which do not satisfy at least one of the inequalities. `Bertini`

makes available a fast polynomial evaluator as another feature independent of homotopy continuation and the numerical irreducible decomposition. Components for handling polynomial evaluation and filtering points have been added to the SA's implementation to fulfill these requirements.

We have emphasized the need to control the number of equations in our polynomial system throughout the previous discussion. Note, however, that the homotopies in Theorems 2.12 and 2.13 solve systems which have $2N - d + 1$ variables and $2N - d + 1$ equations no matter what nonnegative d is chosen. The numerical algebraic geometry literature emphasizes the need for randomization in the case of non-square systems. Randomization of the MinPoint and CritPoint homotopies receives much less attention in [14, 16], which introduce them. Several initial attempts at configuring the SA did not use randomization, and correspondingly produced undesirable positive dimensional solutions sets for MinPoint. There is an error in the paper [25] which complicates identifying the correct number of equations for randomization. The paper reports that the polynomial system for cyclooctane contains 20 polynomials rather than 25. Several weeks of work on this example were spent consulting difficulties related to this discrepancy. The authors have confirmed that 25 is the intended number of equations.

One of the chief practical difficulties which arises in this example is that deriving expanded versions of the Cayley-Menger determinant equations requires extensive work with a computer algebra system. The resulting MinPoint and CritPoint homotopy configurations consist of 23 equations in 23 variables. Specialized scripts for generating the initial polynomial system, randomizing to the correct number of equations, and formatting the equations for use in the SA have been written. They are available with the other example code.

Apart from configuring the SA itself, supercomputing facilities demand their own specialized configurations to run jobs. Subsequent waiting times and interruptions in service at available supercomputing facilities have delayed the computation of results. Due to the difficulties in configuring this example and the supercomputing delays, sampling and TDA results are unfortunately not available for this report.

Chapter 5

Conclusion and Future Directions

In this dissertation we develop and implement theoretically sound and computationally tractable methods for analyzing real algebraic varieties using topological data analysis. Perhaps surprisingly, the combination of NAG and TDA theory in Chapter 2 shows that these methods can lead in some cases to computational proofs exhibiting the existence of homology features. While the primary motivation for creating the dense sampling algorithm in Chapter 3 is producing input for TDA methods, well-distributed and minimal samplings of varieties can serve as good input to most data analysis methods. The results in Chapter 4 show that combining sampling and TDA can effectively produce information about a variety's topology and geometry, even when theoretical certainty is not available. They also show the computational cost of taking an adaptive approach to sampling. Even in the very first simple example, the SA took nearly six minutes to derive a minimized sampling for use with TDA. There are several potential avenues for subsequent work in this area, both by improving the algorithm from Chapter 3, and by using the algorithm to apply TDA to varieties.

The heuristics in Chapter 3 for minimizing samples have proven effective, but there are other potential routes for accomplishing the task of sample minimization. Instead of taking care "up front" to avoid adding unnecessary sample points, the SA could instead first collect a larger sampling with less aggressive or no skipping. Manifold learning tools like `ISOMAP` [27] could be combined with MinPoint to estimate new, more efficiently distributed, sample points. A minor adjustment allows the sampling algorithm to find an ϵ-sampling of $\mathcal{V}_{\mathbb{R}}(f) \cap X$ where X is

the union of a finite number of boxes, rather than a single overall bounding box. The SA could potentially speed up finding samples for complicated spaces by first running a faster sampling strategy that does not minimize the sample, followed by executing Chapter 3's stricter strategy with heuristics to minimize the sampling. This initial input can come from Chapter 3's algorithm, or from another source.

Finally, the initial TDA approach presented in this dissertation can be refined with additional methods from NAG to provide richer information about the geometry of real varieties. Recall that for a variety V the set of singular points is a proper subvariety of V. That subvariety might itself contain points which are singular points relative to the subvariety, and so on. In [17], the authors define *isosingular sets* to be sets of points in V which share a singularity structure, and provide methods for efficiently deriving equations that define these sets. Combining these methods with sampling, TDA, and other data analysis methods would provide rich geometric information about the variety V.

References

[1] Daniel J. Bates, Jonathan D. Hauenstein, Andrew J. Sommese, and Charles W. Wampler. Bertini: Software for numerical algebraic geometry. Available at bertini.nd.edu with permanent doi: dx.doi.org/10.7274/R0H41PB5.

[2] Daniel J Bates, Jonathan D Hauenstein, Andrew J Sommese, and Charles W Wampler. *Numerically solving polynomial systems with Bertini*, volume 25. SIAM, 2013.

[3] Ulrich Bauer, Michael Kerber, and Jan Reininghaus. Dipha (a distributed persistent homology algorithm). Software available at `https://github.com/DIPHA/dipha`.

[4] Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger. The r*-tree: an efficient and robust access method for points and rectangles. In *ACM SIGMOD Record*, volume 19, pages 322–331. Acm, 1990.

[5] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[6] Peter Bubenik and Jonathan A Scott. Categorification of persistent homology. *Discrete & Computational Geometry*, 51(3):600–627, 2014.

[7] Gunnar Carlsson. Topology and data. *Bulletin of the American Mathematical Society*, 46(2):255–308, 2009.

[8] Frédéric Chazal, David Cohen-Steiner, Marc Glisse, Leonidas J Guibas, and Steve Y Oudot. Proximity of persistence modules and their diagrams. In

Proceedings of the twenty-fifth annual symposium on Computational geometry, pages 237–246. ACM, 2009.

[9] David Cohen-Steiner, Herbert Edelsbrunner, and John Harer. Stability of persistence diagrams. *Discrete & Computational Geometry*, 37(1):103–120, 2007.

[10] David Cox, John Little, and Donal O'shea. *Ideals, varieties, and algorithms*, volume 3. Springer, 1992.

[11] DF Davidenko. On a new method of numerical solution of systems of nonlinear equations. In *Dokl. Akad. Nauk SSSR*, volume 88, pages 601–602, 1953.

[12] Vin De Silva and Robert Ghrist. Coverage in sensor networks via persistent homology. *Algebraic & Geometric Topology*, 7(1):339–358, 2007.

[13] Robert Ghrist. Barcodes: the persistent topology of data. *Bulletin of the American Mathematical Society*, 45(1):61–75, 2008.

[14] Zachary A Griffin and Jonathan D Hauenstein. Real solutions to systems of polynomial equations and parameter continuation. *Advances in Geometry*, 15(2):173–187, 2015.

[15] Allen Hatcher. *Algebraic Topology*. Cambridge University Press, 2002.

[16] Jonathan D Hauenstein. Numerically computing real points on algebraic sets. *Acta applicandae mathematicae*, 125(1):105–119, 2013.

[17] Jonathan D Hauenstein and Charles W Wampler. Isosingular sets and deflation. *Foundations of Computational Mathematics*, 13(3):371–403, 2013.

[18] Fritz John. Extremum problems with inequalities as subsidiary conditions. In *Traces and emergence of nonlinear programming*, pages 197–215. Springer, 2014.

[19] William Karush. *Minima of functions of several variables with inequalities as side constraints*. PhD thesis, Masters thesis, Dept. of Mathematics, Univ. of Chicago, 1939.

[20] H. W. Kuhn and A. W. Tucker. Nonlinear programming, 1951.

[21] Shawn Martin, Aidan Thompson, Evangelos A Coutsias, and Jean-Paul Watson. Topology of cyclo-octane energy landscape. *The journal of chemical physics*, 132(23):234115, 2010.

[22] John Milnor. *Singular Points of Complex Hypersurfaces*. Number 61. Princeton University Press, 1968.

[23] Nina Otter, Mason A Porter, Ulrike Tillmann, Peter Grindrod, and Heather A Harrington. A roadmap for the computation of persistent homology. *arXiv preprint arXiv:1506.08903*, 2016.

[24] Jose A. Perea and John Harer. Sliding windows and persistence: An application of topological methods to signal analysis. *Foundations of Computational Mathematics*, 15(3):799–838, 2015.

[25] Josep M Porta, Lluis Ros, Federico Thomas, Francesc Corcho, Josep Cantó, and Juan Jesús Pérez. Complete maps of molecular-loop conformational spaces. *Journal of computational chemistry*, 28(13):2170–2189, 2007.

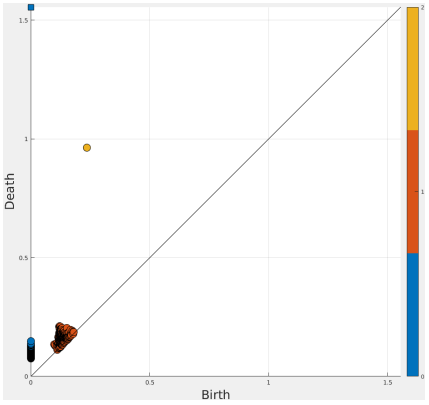
[26] Andrew Sommese and Charles Wampler. *The Numerical solution of systems of polynomials arising in engineering and science*, volume 99. World Scientific, 2005.

[27] Joshua B Tenenbaum, Vin De Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

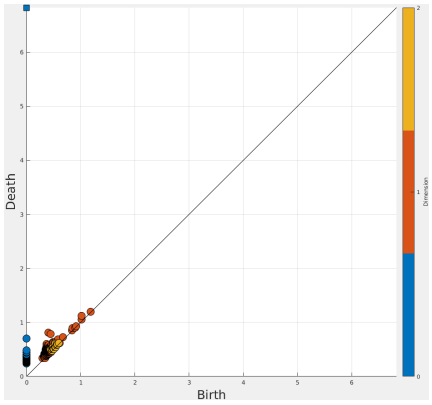
[28] Hassler Whitney. Elementary structure of real algebraic varieties. In *Hassler Whitney Collected Papers*, pages 456–467. Springer, 1992.

[29] Afra Zomorodian and Gunnar Carlsson. Computing persistent homology. *Discrete & Computational Geometry*, 33(2):249–274, 2005.

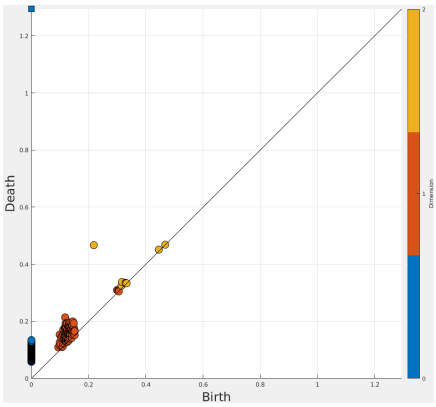
Appendix A: Persistence diagrams for quartics



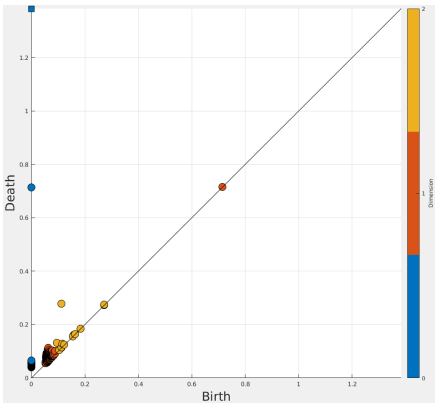
V_1



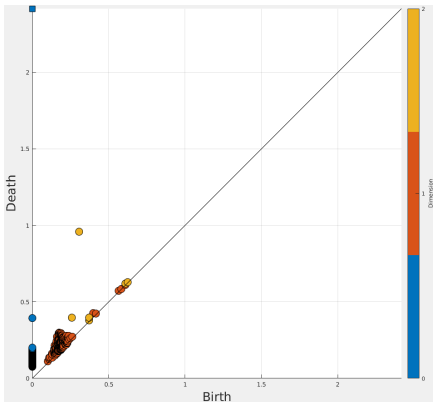
V_2



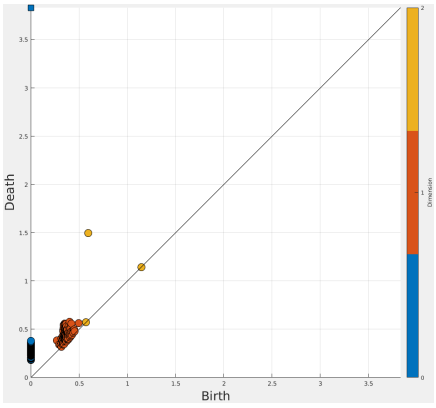
V_3



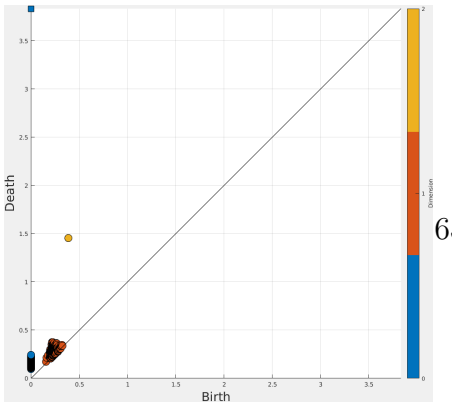
V_4



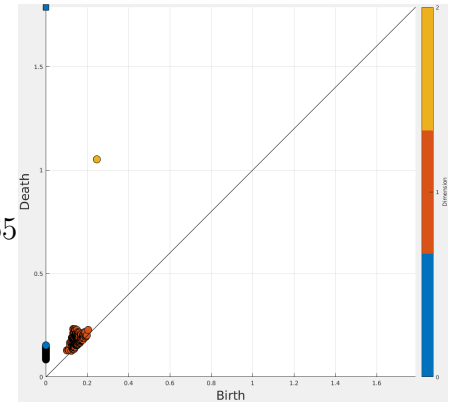
V_5



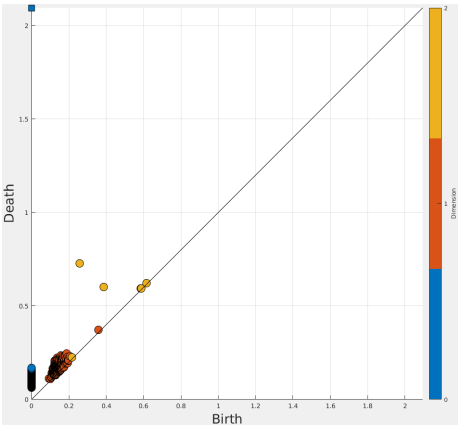
V_6



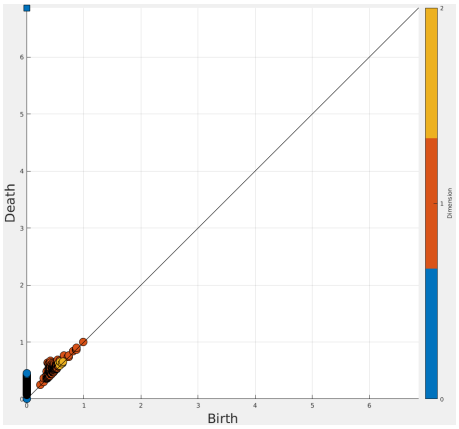
V_7



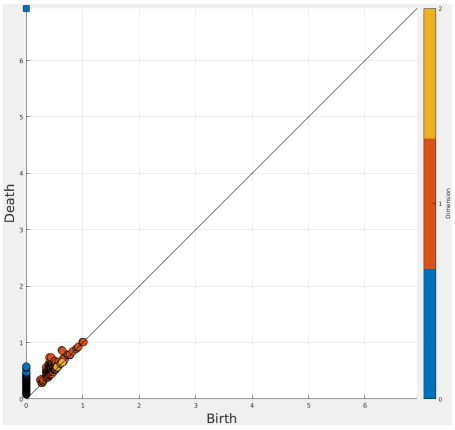
V_8



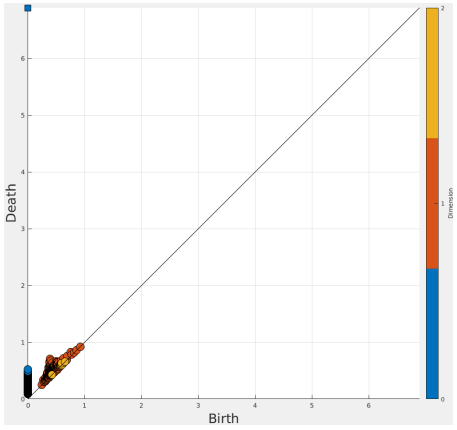
V_9



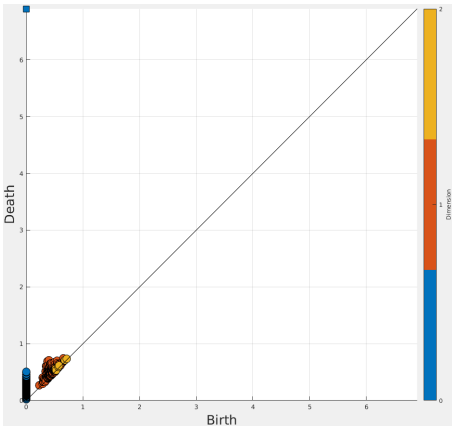
V_{10}



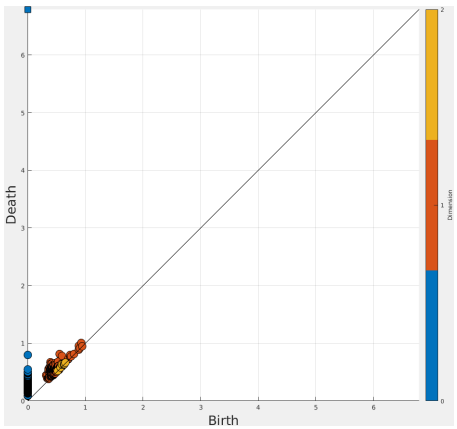
V_{11}



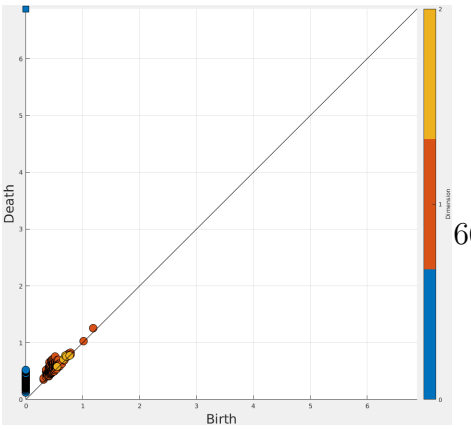
V_{12}



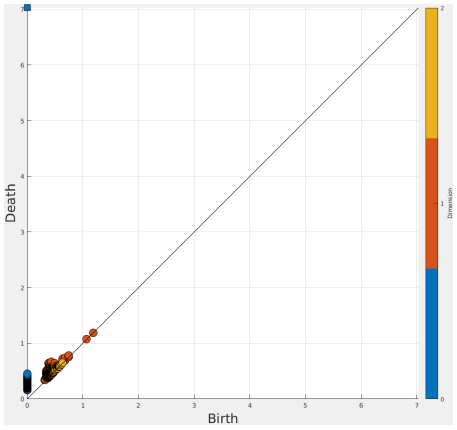
V_{13}



V_{14}



V_{15}



V_{16}