Computing Science and Mathematics
University of Stirling

Evaluating PRNGs using Topological Data Analysis

Zoltan A. Kocsis

Submitted in partial fulfilment of the requirements for the degree of
Bachelor of Science in Computing Science and Mathematics

April 2016

Abstract

The aim of Persistent Homology is to quantify the connectivity of a point cloud the same way homology groups and Betti numbers quantify the analogous properties of topological spaces. We provide an introduction to Persistent Homology, and apply the theory to create tests for distinguishing between random and pseudo-random sequences. The new tests provide versatile - and general - alternatives to well-known tests for pseudo-randomness, including the spectral test.

Attestation

I understand the nature of plagiarism, and I am aware of the University’s plagiarism policy.
I hereby attest that this thesis is entirely my own work and that all additional sources of information have been duly cited.

Signature: **Date:**

Acknowledgements

I would like to thank the following people and organisations.

- **Dr. Marwan Fayed**, my thesis supervisor, for all his help.
- **Dr. Mads Haahr** of RANDOM.ORG, for providing us with the true random numbers I needed to validate my results.
- **My family and friends**, for their love and support.

Ezt a dolgozatot nagyapámnak ajánlom.

Contents

Chapter 1

Introduction

Pseudo-random number generators (PRNGs) are deterministic algorithms that generate sequences of numbers intended to mimic properties of truly random sequences.
Since truly random sequences cannot be produced by deterministic machines, many fields of Computer Science (including cryptography, Monte Carlo methods in Numerical Analysis, scientific and engineering simulations, and Search-based Optimization) rely on *pseudo-random sequences* emitted by PRNGs.
Hungarian-American mathematician John von Neumann invented the first PRNG algorithm in 1946 [14], years before any electronic computers were built [10]. Unfortunately, his method turned out to be a poor source of random numbers due to the presence of short repetitive sequences.
Another early generator, the infamous RANDU algorithm developed by IBM and included in as part of their Scientific Subroutine Package [7], produces output that appears uniformly random, actually displays extreme regularity (”falling in the planes”) when sampling a space of 3 or more dimensions. This behavior makes RANDU unsuitable for any scientific application.
Due to these incidents, statistical tests have been developed to compare the properties of pseudo-random sequences with properties of truly random ones: a hypothesis test can be defined for any statistic with a well-defined distribution under the null hypothesis “the sequence is random”. A battery of tests is necessary, since one test can detect only a single type of departure from the null hypothesis. The U.S. National Institute of Standards and Technology (NIST) maintains the most widely used general-purpose statistical test suite for evaluating random and pseudo-random number generators [3].
Statistical tests have two possible outcomes: accepting or rejecting the null hypothesis. Theoretical results can give us more understanding about our PRNGs than these statistical tests. One such result, the *spectral test* of Marsaglia [15] uses the fact that all Linear Congruential generators (the family of RNGS containing RANDU) has the ”falling in planes” defect in high dimensions. According to Knuth [14], all good Linear Congruential PRNGs pass the spectral test and all the ones *known to be bad* actually fail it. Due to the reliance on the plane structure, the spectral test does not generalize to other types of PRNGs.

1.1 Objectives

The last ten years saw the emergence of new data analysis tools coming from algebraic topology, among them *persistent homology* [6].
Persistent homology quantifies the connectivity of a point cloud the same way *homology groups* and Betti numbers quantify the analogous properties of topological spaces. This allows us to computationally identify connected components, loops and even higher dimensional “holes” present in a point cloud. [12]
The plane structures of Linear Congruential generators can be seen as topological artifacts. It is conceivable that other families of PRNGs also have characteristic topological signatures in higher dimensions. Our objectives are to contrast the persistent homology of true random numbers and PRNG output, and to create new tests for random number generators based on topological data analysis. These topological tests should inherit benefits of the statistical approach (i.e. working across different families of generators), while retaining the power of the more theoretical tests such as the spectral test.

1.2 Previous Work

The history of pseudo-random number generators starts with the 1946 *middle-squaring method* of Hungarian-American mathematician John von Neumann [14]; by 1951, the method was widely used in the ENIAC (with a slight improvement proposed by Ulam) [25]. The generator had several flaws: the maximal period of middle-squaring generators is quite low, many seed values lead to short repetitive sequences, and zeros in the seed are propagated permanently.
The most popular family of random number generators, the so-called Linear Congruential Generators, were introduced in 1951 by Lehmer [24]. The infamous RANDU algorithm of IBM [7] belongs to this family. RANDU produces output that appears uniformly random, even though consecutive values are correlated. When the output of Randu is plotted in three dimensions, all the points lie on 15 parallel planes. In 1963, Marsaglia demonstrated that every Linear Congruential Generator has the same problem in some dimension [15]: in some dimension n-tuples outputted by a Linear Congruential Generator lie on parallel hyperplanes. Marsaglia's result led to the powerful *spectral test* [14] for measuring the quality of Linear Congruential Generators.
CERN's need for reliable random number generators prompted the creation of the McGill University Random Number Package in 1970 (the generator itself is sometimes known as the "super-duper"). It turned out to be sensitive to choice of seed: only some seeds were suitable, which made it difficult to run multiple simulations in parallel [20].
The generators RANMAR and RANLUX were designed so that every different seed provides a disjoint sequence. These are extremely high quality generators with no known serious flaws [13].
The most widely-used random number generator is the Mersenne Twister [18]. This generator has an inexhaustible period ($2^{19997} - 1$) and very good statistical distribution properties, although it does suffer from seed issues similar to those of the "super-duper".
The first statistical test suite for evaluating PRNG quality was the 1995 DieHard suite of Marsaglia. The tests of the suite examine the distribution of numbers, the correlation between successive elements, periodicities, and reliability on Monte Carlo tasks. A successor of the DieHard suite, the NIST suite, consists of 15 tests selected and maintained by the National Institute of Standards and Technology (NIST) [3].
Topological data analysis (TDA) is the collective name for a family of data analysis methods developed in the 2000s for inferring global properties of high-dimensional data. The most successful of these tools is a technique of algebraic topology called *persistent homology* [6].
The first applications of Algebraic Topology in Computer Science were in areas such as surface reconstruction [1]. The first NSF workshop on Computational Topology was held in 1999.
The idea of *persistence* appeared simultaneously in Frosini's work on point cloud clustering (size theory) [8] and Edelsbrunner's 2002 article introducing persistent homology [9]. Since its introduction, persistent homology has been used successfully in protein analysis, image processing, text analysis and other fields. The present work relies heavily on the stability and recovery results of Cohen-Steiner, Edelsbrunner and Harer [5], which relate persistent homology to ordinary (simplicial and singular) homology.

Chapter 2

Mathematical Foundations

2.1 Simplicial Homology

The study of homology was initiated by Henri Poincare in 1895, for the purpose of mathematically classifying topological spaces according to their connectivity properties [21]. The numerous homology theories of mathematics generalize Poincare's techniques to various settings. The theory of persistent homology, which we will use to analyze the output of random number generators, can be seen as one such generalization, generalizing from the setting of topological spaces to the setting of point clouds.
In this section, we give a self-contained introduction to *simplicial homology*, and cover the bare minimum of *singular homology*. Building on that, Section 2.3 introduces *persistent homology*. This section presumes familiarity with some abstract algebraic constructions, including quotient groups. For an overview, the reader is referred to the relevant chapters of Artin's book [2]. For a more in-depth look at singular homology, see Massey's treatment [17].

2.1.1 Homology of Graphs

In this introductory section, we explore topological properties of graphs from an algebraic perspective. In what follows, let $\mathcal{G}$ be an undirected graph with node set N and edge set E.

Definition 1. Two nodes $n_1, n_2 \in N$ belong to the same connected component if there is a path in $\mathcal{G}$ with end-points n_1 and n_2. If that's the case, we write $n_1 \sim n_2$. Equivalence classes of nodes w.r.t. the equivalence relation $\sim$ are called *connected components* of the graph.

We will now give a purely algebraic description of connected components. The algebraic analogue of a "path" is called an "edge chain". We also define node chains, which represent sets of nodes.

Definition 2. An *edge chain* is a formal sum of edges of $\mathcal{G}$, subject to the following relations: for all edges $e_1, e_2 \in E$, $e_1 + e_2 = e_2 + e_1$ and $e_1 + e_1 = 0$.

Definition 3. A *node chain* is a formal sum of nodes of $\mathcal{G}$, subject to the following relations: for all nodes $a, b \in N$, $a + b = b + a$ and $a + a = 0$.

Intuitively, an edge chain can be imagined as a sequence of clicks in a graph editing program: the first click selects an edge, the second click unselects it. Thus, the edge chain $e_1 + e_3 + e_2 + e_3$ represents the following sequence of actions:

1. Click on the edge e_1.
2. Click on e_3.
3. Click on e_2.
4. Click on e_3 again.

The second click on e_3 unselects it, so this sequence of actions selects e_1 and e_2 only. Algebraically, we write $e_1 + e_3 + e_2 + e_3 = e_1 + e_2$. See Figures 2.1 and 2.2 for examples of edge chains.

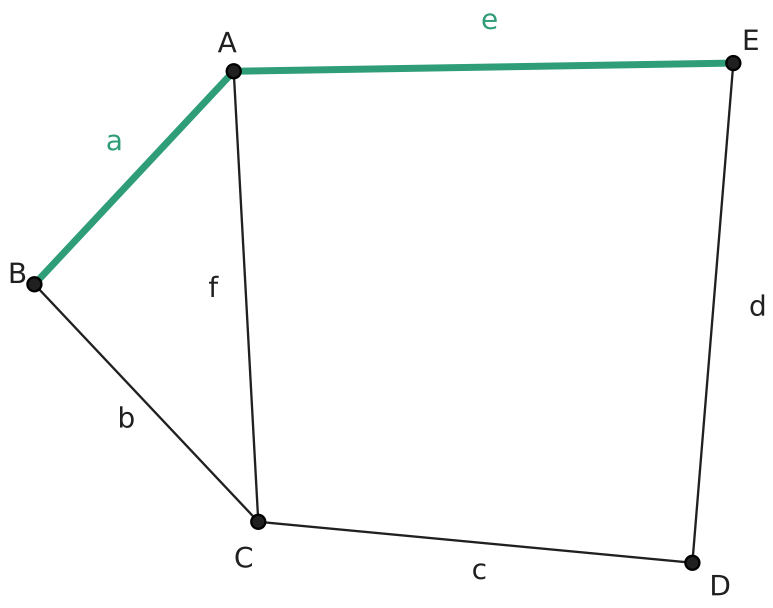

Figure 2.1: The edge chain $a + e$ corresponds to the highlighted edges.

Proposition 1. *The set of all edge chains C_1 (resp. node chains C_0) forms a vector space over the finite field $\mathbb{F}_2$.*

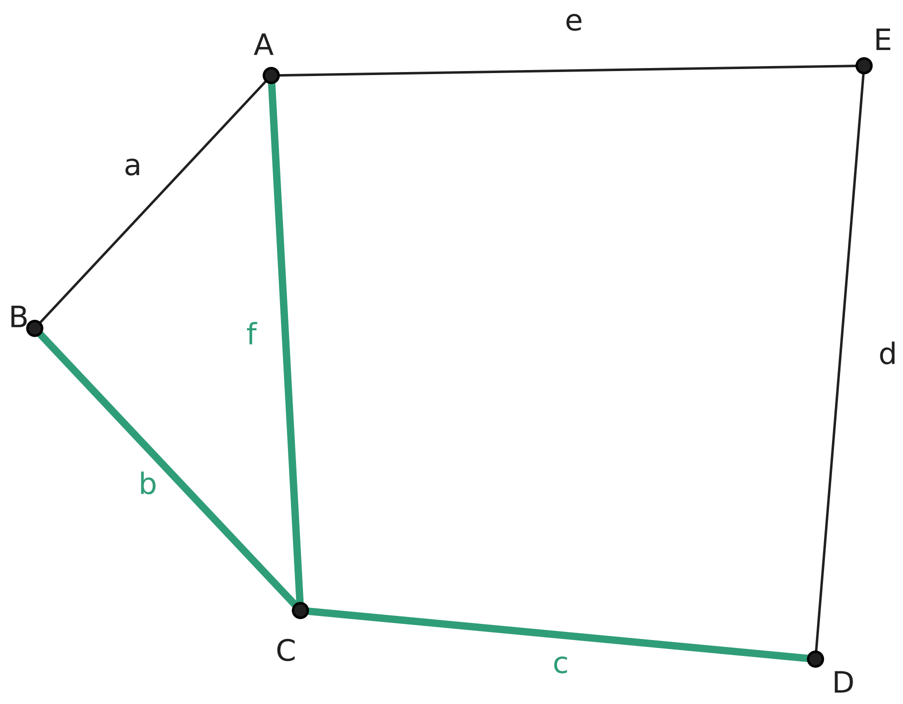

Figure 2.2: The edge chain $b + c + f$ corresponds to the highlighted edges. Not all edge chains are paths.

Intuitively, the vectors of C_1 are bit vectors representing selected edges. For example, in a graph with five edges, the bit vector 01101 could represent the chain $e_2+e_3+e_5$. The addition operation of C_1 is "exclusive or", while the multiplication operation is "and".
Paths had endpoints, their algebraic analogues will have boundaries.

Definition 4. The boundary map $\partial_1 : C_1 \to C_0$ is defined as the unique linear transformation that sends each edge to the sum of its endpoints.

For an example of a boundary, see Figure 2.3. The next theorem shows that being connected by an edge chain is just as good as being connected by a path.

Proposition 2. *There is a path with endpoints A and B if and only if there is an edge chain with boundary $A + B$.*

Proof. ($\Rightarrow$) Assume there is a path $e_1, \dots, e_k$ with endpoints A and B. Then $\sum_{i=1}^{k} e_i$ is clearly an edge chain chain with boundary $A + B$.

($\Leftarrow$) Assume there is an edge chain c with boundary $A+B$. Proceed by induction on the number of edges in c.
Base case: c consists of one edge, and this edge has to have endpoints A and B.
Inductive case: there's at least one edge e of the chain with endpoint A. Let C denote the other endpoint. The chain $c+e$ has fewer edges than the chain c, and $\partial(c+e) = C+B$. By induction, there's a path from C to B. $\square$

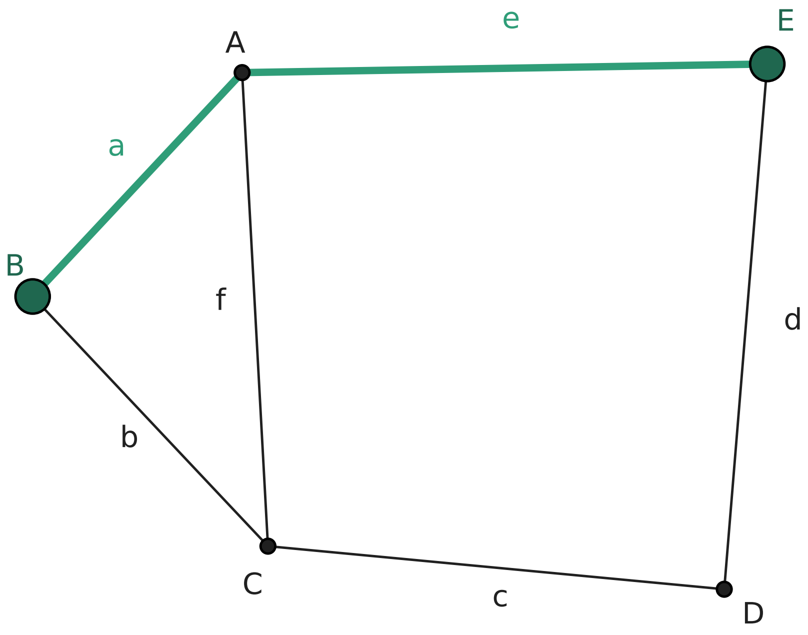

Figure 2.3: The boundary of the edge chain $a+e$ is $(A+B)+(A+E) = B+E$.

As a corollary of Proposition 2, we obtain a purely algebraic way of counting connected components.

Proposition 3. *The number of connected components of a graph is given by* $\dim C_0/\operatorname{im}\partial_1$ *where* C_1 *and* C_0 *are the associated edge and node chain vector spaces, and* $\partial_1 : C_1 \to C_0$ *is the boundary map.*

Proof. By Proposition 2, if $A+B \in \operatorname{im}\partial_1$ then $A \sim B$ and vice versa. $\square$

The properties of the boundary map are related to other topological invariants of the graph as well. The next theorem counts the faces of a graph purely algebraically.

Proposition 4. *The number of faces of a planar graph is given by* $\dim \ker \partial_1$.

Proof. First of all, notice that every cycle is an edge chain with boundary 0, i.e. a vector in $\ker \partial_1$. Draw the graph in the plane. In the drawing, f can be uniquely identified with its bounding cycle c_f. What's more, any other cycle c must surround a set of faces F, which means that it can be written as $c = \sum_{f \in F} c_f$. In other words, the cycles corresponding to faces form a *basis* of $\ker \partial_1$, and therefore $\dim \ker \partial_1$ is the number of faces. $\square$

Aside: since every basis of a finite-dimensional vector space has the same number of elements, we can immediately conclude that all plane drawings of a graph have the same number of faces.

Definition 5. The vector spaces $H_0 = C_0/\mathrm{im}\, \partial_1$ and $H_1 = \ker \partial_1$ are called (respectively) the zeroth and first homology groups of the graph $\mathcal{G}$.

2.1.2 Homology of Simplicial Complexes

In this section, we generalize homology from graphs to simplicial complexes.

Definition 6. An abstract *n-simplex* is a set of $n+1$ elements. A *simplex* is an n-simplex for some $n \in \mathbb{N}$.

Definition 7. The boundary set of an n-simplex is the collection of its n-element subsets.

Definition 8. An abstract *simplicial complex* is a set of simplices closed under inclusion.

Geometrically, a 0-simplex can be thought of as a point, a 1-simplex as a line segment, a 2-simplex as a triangle, a 3-simplex as a tetrahedron and so on for higher dimensions. A simplicial complex is an object "glued together" from simplices. See Figure 2.4 for an example.
Every graph is a simplicial complex consisting of 0- and 1-simplices.
The definition of homology groups for simplicial complexes is similar to the definition for graphs, except for two modifications.

1. Instead of faces, H_1 is going to count two-dimensional *holes*. For example, the simplicial complex of Figure 2.4 has only one hole, since the other candidate is filled by "filled" by the 2-simplex on the left.

2. It will make sense to define higher homology groups $H_2, H_3, \ldots$ as well.

In the following, let $\mathcal{G}$ be a simplicial complex.

Definition 9. An *n-chain* is a formal sum of n-simplices of $\mathcal{G}$, subject to the following relations: for all n-chains a, b, we have $a+b = b+a$ and $a+a = 0$. The group of n-chains is denoted by C_n.

Definition 10. The boundary map $\partial_{n+1} : C_{n+1} \to C_n$ is the unique linear transformation that assigns to every $(n+1)$-simplex the formal sum of its boundary set.

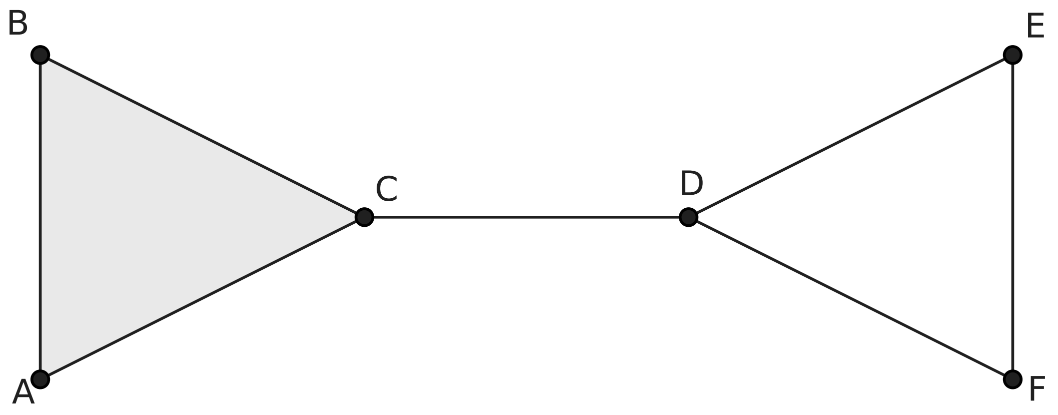

Figure 2.4: A simplicial complex consisting of six 0-simplices, seven 1-simplices and a single 2-simplex.

Definition 11. The nth *homology group* of the simplicial complex $\mathcal{G}$ is defined as

$$H_n = \ker \partial_n / \operatorname{im} \partial_{n+1}$$

The nth *Betti number* of the simplicial complex $\mathcal{G}$ is defined as

$$\beta_n = \dim H_n$$

The following theorem ensures that the quotient group above is well-defined.

Proposition 5. *Every boundary is a cycle.*

Proof. Let S be any $(n+1)$-simplex. By definition,

$$\partial_{n+1} S = \sum_{x \in S} S \setminus \{x\},$$

$$\partial_n \partial_{n+1} S = \sum_{y \in S \setminus \{x\}} \sum_{x \in S} S \setminus \{x, y\}$$

The second sum is zero because every term occurs exactly twice: first in the form $S \setminus \{a, b\}$ and then as $S \setminus \{b, a\}$. Consequently, $\partial_{n+1} S$ has no boundary, i.e. it is a cycle. □

Intuitively, β_0 is the number of connected components, and for $n > 0$, β_n counts the number of unfilled $(n+1)$-dimensional holes in the simplicial complex. Notice that all of these quantities can be computed straightforwardly, using the standard algorithms of linear algebra.

2.2 Singular Homology

It's time to extend homology to arbitrary topological spaces. The idea is to "draw" simplices on the space. Consider the disk (Fig. 2.5) and the annulus (Fig. 2.6). Every triangle drawn on the disk is the boundary of some 3-simplex drawn on the disk. However, there are triangles drawn on the annulus that are not boundaries of any 3-simplex. The reader should note that the word "triangle" is used in the topological sense here, i.e. anything homeomorphic to a triangle is considered to be a triangle.

By rigorously defining "drawing a simplex on a space", we can obtain homology groups for arbitrary topological spaces. Let T be a topological space, and let Δ^n denote the geometric realization of an n-dimensional simplex. Every point of Δ^n has to be drawn somewhere: thus, any method of drawing Δ^n on T will define a function $f : \Delta^n \to T$ which, given a point $p \in \Delta^n$, tells us the point's location $f(p) \in T$ on the drawing. What's more, nearby points must have nearby locations on the drawing. These considerations suggest the following

Definition 12. Let T be a topological space. Let Δ^n denote an n-simplex. A *singular n-simplex* on T is a continuous map $f : \Delta^n \to T$.

Intuitively speaking, a *singular n-simplex* is just a method of drawing an n-simplex on a given space[1]. Definition 12 is quite general, which means that we allow degenerate drawings (hence the term "singular"). For example, constant functions are singular n-simplices for all $n \in \mathbb{N}$.

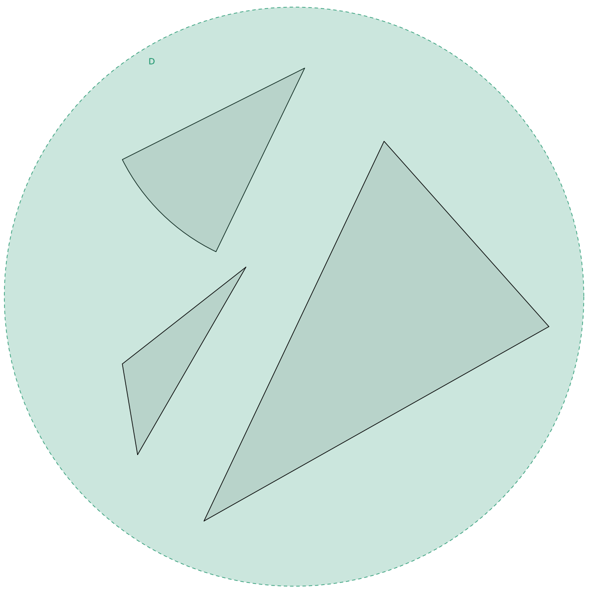

Figure 2.5: Every triangle on the disk is the boundary of some 3-simplex drawn on the disk.

[1]This is similar to Complex Analysis, where we identify *contours* with their parametrizations, i.e. continuous maps $f : [0, 1] \to \mathbb{C}$.

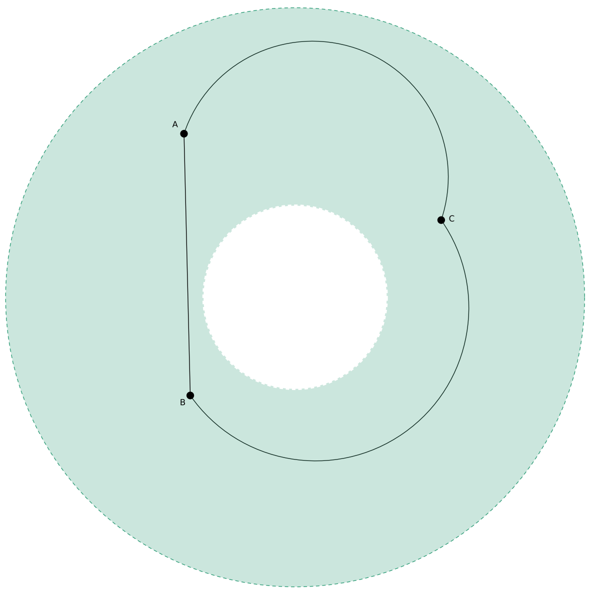

Figure 2.6: The triangle ABC drawn on the annulus T is not the boundary of any 3-simplex drawn on the annulus.

There is nothing surprising about the following definitions.

Definition 13. Let T be an arbitrary topological space. A *singular n-chain* is a formal sum of finitely many singular n-simplices on $\mathcal{T}$, subject to the following relations: for all singular n-chains a, b, we have $a + b = b + a$ and $a + a = 0$. The group of singular n-chains is denoted by $C_n(T)$. The boundary map $\partial_{n+1} : C_{n+1}(T) \to C_n(T)$ is the unique linear transformation that assigns to every singular $(n + 1)$-simplex the formal sum of its boundary simplices. The *singular homology groups* and the *Betti numbers* are defined the obvious way:

$$H_n(T) = \ker \partial_n / \operatorname{im} \partial_{n+1}$$

$$\beta_n(T) = \dim H_n(T)$$

Unfortunately, there are infinitely many different ways of drawing simplices on spaces. Therefore, the chain groups are infinite dimensional vector spaces[2]. Therefore, in the present work, applications of singular homology are restricted to proofs; the computations themselves are carried out using simplicial techniques. When both are defined, the singular homology groups and the simplicial homology groups are identical. For a proof, see [17].
The following is our last proposition about singular homology. Its importance will become apparent later.

[2]For well-behaved spaces, the homology groups themselves are finite-dimensional.

Definition 14. Every continuous function $f : T_1 \to T_2$ gives rise to a corresponding linear transformation $H_n(f) : H_n(T_1) \to H_n(T_2)$ defined by the equation

$$H_n(f)(x + \operatorname{im} \partial_{n+1}) = f \circ x + \operatorname{im} \partial_{n+1}$$

Proposition 6. *Given two continuous functions $f : T_1 \to T_2$, $g : T_2 \to T_3$, the equalities*

$$H_n(g \circ f) = H_n(g) \circ H_n(f)$$

$$H_n(f \circ \mathrm{id}) = H_n(f)$$
$$H_n(\mathrm{id} \circ f) = H_n(f)$$

hold.

Proof. Immediate from Definition 14. □

2.2.1 Digression: Integral Homology

The theory developed in the previous sections is just one possible form of homology, where the coefficients come from the finite field $\mathbb{F}_2$. This is a slightly unusual choice, since *integral homology*, where simplices are oriented and chains have integer coefficients, provides more information about topological spaces. In particular, studying the *torsion subgroups* of the integral homology groups allow us to distinguish twisted spaces from their regular counterparts (e.g. the Klein bottle from the torus). Extending persistent homology to deal with torsion coefficients is an open problem [11].

2.3 Persistent Homology

This section presents a method for deducing the homology of a closed set $\mathfrak{M} \subseteq \mathbb{R}^n$ from a finite sample of points $\mathfrak{p} \subseteq \mathfrak{M}$: *persistent homology*.
Given a point sample $\mathfrak{p}$, we can approximate the original set M by drawing an open ball of radius ε around each point of $\mathfrak{p}$ (see Fig. 2.7). To recover the homology of $\mathfrak{M}$, we have to consider how the homology groups of the approximations *vary* with the parameter ε. Definition 15 is intended

to capture the notion of a topological object varying with some real parameter.

Definition 15. A *diagram* F consists of:

- a family of topological spaces $F(x)$, one for each real number x,
- and continuous transition functions $F_a^b : F(a) \to F(b)$, one for each pair of real numbers $a \leq b$,

such that

- $F_a^a(x) = x$ for all $a, x \in \mathbb{R}$,
- and $F_b^c(F_a^b(x)) = F_a^c(x)$ for all $x \in \mathbb{R}$ and $a \leq b \leq c$.

The persistent homology of the object is the homology that is preserved by transition functions. Each transition will define a persistent homology group.

Definition 16. Given a diagram of topological spaces F, the *nth persistent homology group between a and b* is defined as $H^n(\mathrm{im} F_a^b)$, and when n is unambigous, denoted by $H_a^b F$.

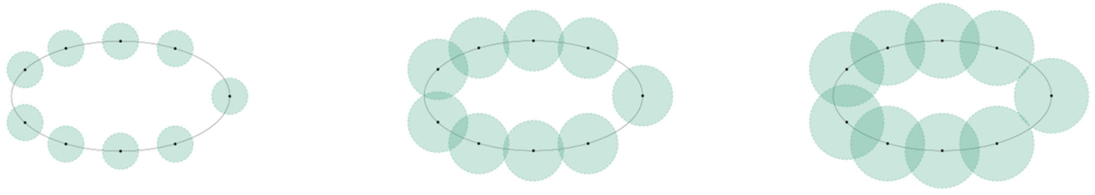

Figure 2.7: Ellipse approximated by a union of open balls around a point sample. The scale parameter ε is the radius of the balls.

2.3.1 Sub-level Diagrams

We can interpret subsets of $\mathbb{R}^n$, with open balls drawn around each point, as diagrams.

Definition 17. The *sub-level diagram* $f_\downarrow$ of a function $f : \mathbb{R}^n \to \mathbb{R}$ is defined as follows:

- for each $\varepsilon \in \mathbb{R}$, $f_\downarrow(\varepsilon) = \{x \in \mathbb{R}^n \mid f(x) < \varepsilon\}$
- the transition functions $f_{\downarrow a}^{\ b}$ are the inclusion maps.

The *sub-level diagram* $S_\downarrow$ of the set $S \subseteq \mathbb{R}^n$ is the sub-level diagram $f_\downarrow$ of the corresponding distance function

$$f(x) = \inf_{s \in S} |x - s|$$

From now on, we freely identify subsets of $\mathbb{R}^n$ with their distance functions and their sub-level diagrams.

2.3.2 Homology Recovery

In this subsection, we prove the recovery theorem of Cohen-Steiner, Edelsbrunner and Harer [5]. This theorem provides sufficient conditions for recovering the homology of a closed set from the persistent homology of a finite sample. Originally, the recovery theorem was stated in terms of bottleneck distance - our proofs are shorter since they rely on the more recent notion of interleavings introduced by Bubenik and Scott [4].
If two diagrams "always stay close", they are *interleaved.* Intuitively, two diagrams should be interleaved if $F(x) \subseteq G(x+\varepsilon) \subseteq F(x+2\varepsilon)$ for all $x \in \mathbb{R}$. In practice, we require a bit more: the transition functions also need to be compatible. This requirement is formalized by Def. 18.

Definition 18. Two diagrams F, G are *ε-interleaved* if $F(x) \subseteq G(x+\varepsilon)$, $G(x) \subseteq F(x+\varepsilon)$ and the transitions are compatible, i.e. $G^{y}_{x+\varepsilon}(F(x)) = F(y+\varepsilon)$ and $F^{y}_{x+\varepsilon}(G(x)) = G(y+\varepsilon)$ for all $x, y \in \mathbb{R}$.

Let us prove that interleaving is the right notion of "distance" for diagrams.

Proposition 7. *If two functions f and g satisfy $|f(x) - g(x)| < \varepsilon$ for all x, then the diagrams $f_\downarrow$ and $g_\downarrow$ are ε-interleaved.*

Proof. The transition functions are inclusion maps, so it suffices to prove that $f_\downarrow(k) \subseteq g_\downarrow(k+\varepsilon)$ and vice versa. For any $x \in f_\downarrow(k)$, we have that $f(x) < k$. Therefore,

$$g(x) < f(x) + \varepsilon < k + \varepsilon,$$

so $x \in g_\downarrow(k+\varepsilon)$. The other direction works identically. □

Proposition 8. *If two functions f and g satisfy $|f(x) - g(x)| < \varepsilon$ for all $x \in \mathbb{R}^n$, then the persistent homology groups satisfy* $\dim H^{b+\varepsilon}_{a-\varepsilon} f_\downarrow \leq \dim H^{b}_{a} g_\downarrow$.

Proof. By Proposition 7, the sub-level diagrams $f_\downarrow$ and $g_\downarrow$ are ε-interleaved, which proves that $f^{a+\varepsilon}_{\downarrow b-\varepsilon} \subseteq g^{a}_{\downarrow b}$. Apply Proposition 6 to the inclusion map to conclude $\dim H^{b+\varepsilon}_{a-\varepsilon} f_\downarrow \leq \dim H^{b}_{a} g_\downarrow$. □

Finally, all the pieces are in place for proving Proposition 9, a theorem which gives explicit bounds on how fine the point sample has to be to guarantee successful recovery of the (singular) homology of a closed set $\mathfrak{M} \subseteq \mathbb{R}^n$ from the persistent homology of a finite set of points $\mathfrak{p} \subseteq \mathfrak{M}$.

Proposition 9. (COHEN-STEINER, EDELSBRUNNER, HARER)
Let $\mathfrak{M}$ be a closed subset of $\mathbb{R}^n$. Let $\mathfrak{p} \subseteq \mathfrak{M}$ be a finite set of points. If there exists $\ell < u \in \mathbb{R}$ such that

1. *every point of $\mathfrak{M}$ is ℓ-close to some point in $\mathfrak{p}$, and*
2. *the diagram $\mathfrak{M}_\downarrow$ is constant on the interval $(0, 4u)$*

then the singular homology of $H\mathfrak{M}$ is the persistent homology of $H_\varepsilon^{3\varepsilon}\mathfrak{p}$ for any $\varepsilon \in (\ell, u)$.

Proof. Choose any ε such that $\ell < \varepsilon < u$. By Condition 1 and Proposition 7, $\mathfrak{M}$ and $\mathfrak{p}$ are ε-interleaved. Proposition 8, along with Condition 2, gives

$$\dim H\mathfrak{M} = \dim H_0^{4\varepsilon}\mathfrak{M} \leq \dim H_\varepsilon^{3\varepsilon}\mathfrak{p} \leq \dim H_{2\varepsilon}^{2\varepsilon}\mathfrak{M} = \dim H\mathfrak{M}$$

and that proves $\dim H\mathfrak{M} = \dim H_\varepsilon^{3\varepsilon}\mathfrak{p}$. $\square$

Notice that the theorem requires that the homology of $\mathfrak{M}$ be known in advance. This will be the case in our applications.

2.3.3 Computation

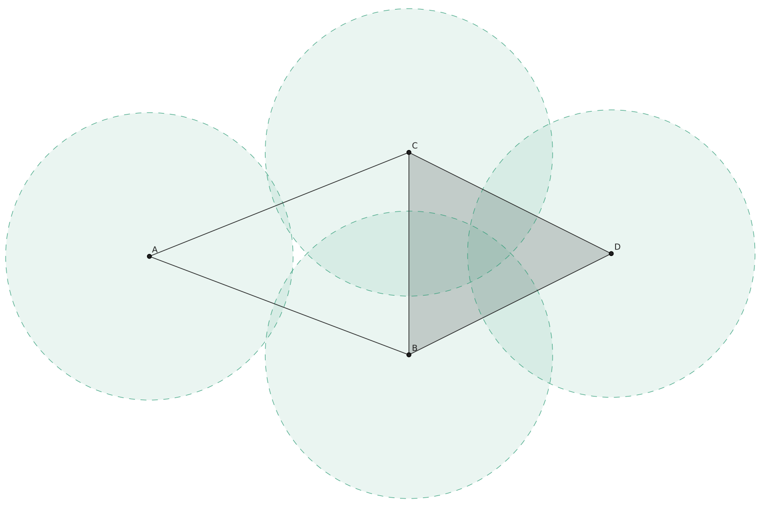

Figure 2.8: Open balls drawn around the points A, B, C, D. The Cech complex is shown in black.

We use version 2.4.4 of the JavaPlex [23] software package for computing the persistent homology of point sets. JavaPlex uses the Cech complex construction (see Figure 2.8) to reduce the computation to the simplicial case, which can be done using linear algebra, along with further optimizations (e.g. the Cech complex itself is bounded by flag complexes). These computational techniques are outside the scope of the thesis: the interested reader is referred to Ghrist's book [12].

Chapter 3

Approach

```
// minimal Java implementation of Randu
public Randu {
  private long seed = 0xFE5327ED;
  public double nextDouble() {
    seed = (seed*65539) % Math.pow(2L,32);
    return Math.abs( (double)seed / modulus );
  }
}
```

The algorithm above is IBM's infamous RANDU [7], a random number generator so ill-behaved that "its very name [...] is enough to bring dismay into the eyes and stomachs of many computer scientists" (Donald Knuth). Amazingly, RANDU is capable of passing some statistical tests for randomness. If RANDU is used to sample random points from the unit square, its output looks random - at least to the naked eye. However, if RANDU is used to sample random points from the unit cube, a topological anomaly reveals itself (Fig. 3.1). This error pattern remained undetected for years. However, once the pattern is found, it is easy to provide a mathematical explanation.

Proposition 10. *Three consecutive outputs of* RANDU *are linearly related.*

Proof. Notice that $65539 = 2^{16} + 3$. Let x_n denote the nth output of RANDU. We have

$$x_{n+2} = (2^{16} + 3)x_{n+1} = (2^{32} + 6 \cdot 2^{16} + 3 + 6)x_n$$

Since RANDU works modulo 2^{32}, the whole expression reduces to

$$x_{n+2} = 6 \cdot (2^{16} + 3)x_n + 9x_n = 6x_{n+1} + 9x_n$$

which is a linear relation between three consecutive outputs. □

It is fortunate that the patterns displayed by RANDU can be detected by visual inspection. Marsaglia [15] proved that every Linear Congruential Generator suffers from similar flaws in some dimension: they are topological artifacts of all Linear Congruential Generators. It is conceivable that other families of PRNGs also have characteristic topological signatures, which may be difficult to detect by visual inspection (either because they are subtle, or because of dimensionality issues). Our preliminary results indicate that this is indeed the case.

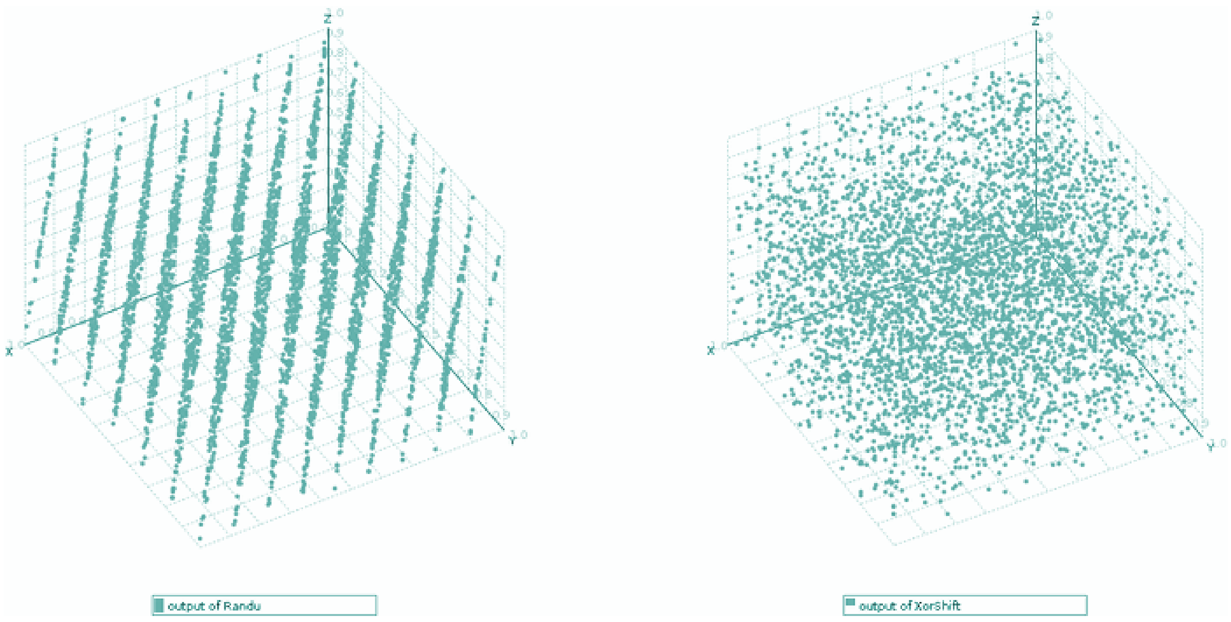

Figure 3.1: *Left*: Output of RANDU in the unit cube - the points lie on 15 parallel planes. *Right*: Output of the XORSHIFT generator - no visible topological anomalies.

In the following pages, we introduce three novel tests for detecting the anomalous behavior of PRNGs, all based on persistent homology. Each of the tests starts with a closed subset $\mathfrak{M}$ of some metric space, uses a PRNG to sample $\mathfrak{M}$, and attempts to recover the homology from the sample. The distribution of failures of the recovery procedure allows us to draw conclusions about the randomness of the sampling procedure.

3.1 Unit Hypercube Homology Test

3.1.1 Purpose

The purpose of the test is to verify that taking k samples of n consecutive outputs of the PRNG approximates a k-point uniform random sample from an n-dimensional hypercube. The effect is similar to, but more general than Marsaglia's spectral test [15], since the generator need not belong to the Linear Congruential family.

3.1.2 Description

Parameters: The dimension of the hypercube d, the size of the point sample n.

Steps:

1. Take an d-dimensional unit hypercube $\mathfrak{M}$. Use the generator to obtain a set $\mathfrak{p}$ of n points from $\mathfrak{M}$.
2. Compute the (zeroth) persistent Betti numbers $\beta_j^i = \dim H_j^i \mathfrak{p}$ with respect to the Euclidean distance for a range of values $j < i$.
3. Calculate the χ^2 test statistic by comparing β against the reference distribution.

Null hypothesis: $\mathfrak{p}$ is a uniform random sample from an n-dimensional hypercube.

3.1.3 Test Statistic

The test statistic follows a χ^2 distribution, measuring how well the observed persistent Betti numbers match the expected Betti numbers under the assumption that the sequence is truly random. The reference distribution is obtained empirically from the control sequences, and an interval of interest is derived below.

3.1.4 Theory

In this section, we derive an interval of interest for this test. The hypercube $\mathfrak{M}$ vacuously satisfies the second condition of the Cohen-Steiner - Edelsbrunner - Harer Theorem (Proposition 9), since the persistent homology diagrams are constant in any interval. Assuming the null hypothesis, $d \leq 3$ and $n \geq 10000$, every point of $\mathfrak{M}$ is 0.08-close to some point in $\mathfrak{p}$ with probability > 0.95. Therefore, $\dim H_{0.08}^{0.24} \mathfrak{p} = 1$ (because the hypercube has only one connected component) and the homology of the hypercube can be recovered from the point sample.

3.2 Matrix Rank Homology Test

3.2.1 Purpose

This test generalizes the NIST [3] Binary Matrix Rank test using persistent homology. The purpose of the test is to verify that there are no unexpected linear dependences between outputs of the sequence.

3.2.2 Description

Parameters: The number of square matrices to use n, the dimension of each matrix d.

Steps:

1. Use the generator to obtain a set $\mathfrak{M}$ of n matrices, each of dimension $d \times d$.
2. Compute the persistent homology groups $H^i_j\mathfrak{M}$ for each $j \leq i \leq d$ with respect to the rank distance, and the corresponding persistent Betti numbers $\beta^i_j = \dim H^i_j\mathfrak{M}$.
3. Compute $\chi^2(\beta)$ and compare against the reference distribution.

Null hypothesis: $\mathfrak{M}$ is a uniform random sample from $\mathbb{F}_2^{d\times d}$.

3.2.3 Test Statistic

The test statistic follows a χ^2 distribution, measuring how well the observed persistent Betti numbers match the expected Betti numbers under the assumption that the sequence is truly random. The expected values are calculated empirically, but their magnitudes can be predicted theoretically.

3.2.4 Theory

Let us prove that the norm $|A| = \operatorname{rank} A$ makes the set of $d \times d$ matrices into a normed vector space. The only non-trivial part is the triangle inequality: $\operatorname{rank}(A+B) \leq \operatorname{rank} A + \operatorname{rank} B$. If a vector belongs to $\operatorname{im}(A+B)$, then it has the form $(A+B)\,v = Av + Bv$ for some $v \in \mathbb{F}_2^d$. Thus, it can be written as the linear combination of a vector in $\operatorname{im} A$ and a vector in $\operatorname{im} B$, proving that $\operatorname{im}(A+B) \subseteq \operatorname{im} A + \operatorname{im} B$. Rank is just the dimension of the image, so the triangle inequality holds. The function $d(x, y) = \operatorname{rank}(x - y)$ is therefore a metric over the set of $d \times d$ matrices. Proposition 11 counts the $n \times k$ matrices of a given rank r. Knowing that allows us to derive the exact distribution of the ranks of random matrices and to make observations about the expected Betti numbers for the persistent homology.

Proposition 11. *Let $N(k,r)$ denote the number of $d \times k$ matrices with rank r for some $d \in \mathbb{N}$. Then the following recurrence relation holds:*

$$N(k+1, r+1) = (2^d - 2^r)N(k,r) + 2^{r+1}N(k, r+1)$$

Proof. There are two ways to construct a d-by-$(k+1)$ matrix of rank $r+1$:

1. Start from a d-by-k matrix M of rank r. Augment M with a row $v \notin \operatorname{span} M$.

2. Start from a d-by-k matrix M of rank $r+1$. Augment M with a row $v \in \operatorname{span} M$.

In a vector space over $\mathbb{F}_2$, we can form 2^n different linear combinations from n lineraly independent vectors, so in the first case we can choose from $2^d - 2^r$ different vectors v, and in the second case we can choose from 2^{r+1} different vectors v. $\square$

As usual, the base cases for the recurrence are trivial, and thus we can pre-compute function N for arbitrary d, k, r. The probability that a uniformly random $d \times d$ matrix has rank r is simply $\frac{N(d,r)}{2^{d \times d}}$. However, our main interest in the homological case is not the distribution of ranks, but the distribution of differences of ranks. Fortunately, we can use Proposition 11 as a stepping stone for that calculation.

Proposition 12. *The probability that* $\operatorname{rank}(A-B) = r$*, where A, B are d-by-d matrices, is $\frac{N(d,r)}{2^{d \times d}}$.*

Proof. The space of matrices $\mathbb{F}_2^{d \times d}$ with matrix addition is a vector space, and so, a fortiori it is a group. Consequently, every element occurs the same number of times in the addition table of $\mathbb{F}_2^{d \times d}$, and so the probability that $\operatorname{rank}(A+B) = r$ is just $\frac{N(d,r)}{2^{d \times d}}$. Finally, we observe that $A + B = A - B$ in $\mathbb{F}_2^{d \times d}$. $\square$

The formula of Proposition 11 can be used to calculate $N(k,r)$ using dynamic programming. We get the following distribution for $d = 64$.

r	64	63	62	< 62
$N(k,r)/2^{r \times r}$	0.29	0.58	0.13	$<$ 0.01

Based on this distribution, we should expect the space to be

- fully connected for rank 64;
- not fully connected for ranks 63 and 62;
- fully disconnected for ranks below 62.

3.3 Subsequence Homology Test

3.3.1 Purpose

The purpose of the test is to verify that there are no unexpected shared bitsequences between the numbers of the sequence.

3.3.2 Description

Parameters: The length of the bitstrings to use k, the number of bitstrings to consider n.

Steps:

1. Use the generator to obtain a set $\mathfrak{M}$ of n bitstrings, each of length k.
2. Compute the persistent homology groups $H_j^i\mathfrak{M}$ for each $j \leq i \leq d$ with respect to the longest-common-subsequence distance, and the corresponding persistent Betti numbers $\beta_j^i = \dim H_j^i\mathfrak{M}$.
3. Compute $\chi^2(\beta)$ and compare against the reference distribution.

Null hypothesis: $\mathfrak{M}$ is a uniform random sample from the set $\{0,1\}^k$ of bitstrings of length k.

3.3.3 Test Statistic

The test statistic follows a χ^2 distribution, measuring how well the observed persistent Betti numbers match the expected Betti numbers under the assumption that the sequence is truly random.

3.3.4 Theory

The longest-common-subsequence distance $d(a, b)$ of two bitstrings a, b is the minimal number of insertion and deletion operations required to transform a into b. We observe that the following edit procedure transforms a into b optimally:

1. Fix an (arbitrary) longest common subsequence s of a and b.
2. Moving from left to right, delete every character in a that does not belong to the subsequence s.
3. Moving from left to right, add every character in b that does not beong to the subsequence s to its correct position.

This proves that d is a metric: clearly $d(a, a) = 0$, and reversing the steps of the optimal edit procedure yields $d(a, b) = d(b, a)$. The triangle inequality holds because the sequential composition of two edit procedures is an edit procedure.

Chapter 4

Evaluation

4.1 Generators

In this section, we introduce the generators that were used to evaluate the performance of the tests proposed in Chapter 3.

4.1.1 Linear Congruential Generators

One of the earliest and most common families of random number generators. A linear congruential generator is a sequence x such that $x_n \in \mathbb{Z}/m\mathbb{Z}$ obeying a recurrence of the form

$$x_{n+1} \equiv ax_n + b \mod m$$

where the constants a, b and the modulus m are charactersitic of the generator. Every linear congruential generator has a lattice structure [15]. We use the follow linear congruential generators in the evaluation:

RANDU: An ill-behaved linear congruential generator from the IBM System/360 Subroutine Package [7]. See Chapter 3 for an extended discussion.

MINSTD: The MINSTD generator is a full period linear congruential generator. It was proposed by Stephen K. Park [19] as a good minimal standard against which other generators can be judged.

Glibc48, java.util.Random: Generators used in the GNU C Library and the Java Class Library.

4.1.2 Inversive Congruential Generators

Inversive congruential generators were designed to overcome the limitations of linear generators. An inversive congruential generator is a sequence x such that $x_n \in \mathbb{Z}/m\mathbb{Z}$ obeying the recurrence

$$x_{n+1} \equiv (ax_n + b)^{-1} \mod m$$

whenever the inverse exists. While inversive generators do not have lattice structures, they sometimes display other topological anomalies.

4.1.3 Feedback Shift Registers

Feedback shift registers are sequences x such that each $x_n \in \mathbb{F}_2^k$ is a bit-vectors of length k, and

$$x_{n+1} = (f(x_n), x_{n1}, x_{n2}, \ldots, x_{n(k-1)})$$

where the *feedback function* $f : \mathbb{F}_2^k \to \mathbb{F}_2$ is characteristic of the generator. We included the follow feedback shift registers in our evaluation:

LFSR2547: A standard 64-bit linear feedback shift register.

Xorshift64: A high-quality 64-bit feedback shift register due to Marsaglia [16]. The smallest known generator that passes all of the DieHard test suite.

4.1.4 Blum-Blum-Shub

The Blum-Blum-Shub family of quadratic pseudorandom number generators was proposed in 1986 by Lenore Blum, Manuel Blum and Michael Shub. A Blum-Blum-Shub generator is a sequence x such that $x_n \in \mathbb{Z}/m\mathbb{Z}$ and

$$x_{n+1} \equiv x_n^2 \mod m$$

where m, a product of two large prime numbers, is characteristic of the generator. As long as output is restricted to one bit per iteration, this generator is cryptographically secure [22]. We used the **BBS2518332913** instance in the evaluation.

4.1.5 Mersenne Twister

A linear generator based on a matrix recurrence (see [18]), the Mersenne Twister family is known for inexhaustibly long periods and good statistical properties. We included the **MT19937** instance, which is currently the most widely-used random number generator.

4.1.6 True Random Sequences

To use as a control, as well as for empirical calculations, we obtained 150 megabytes of true random data from the **Random.org** archive[1].

4.2 Proof-of-Concept

To demonstrate that our algorithms are, in principle, capable of identifying topological anomalies, we ran the Unit Hypercube Homology Test in dimension 2 on "toy" linear and inversive congruential generators with the following parameters: $(a = 61, b = 2, m = 256)$. These generators have short, exhaustible periods, and clearly visible topological anomalies (Figure 4.1).

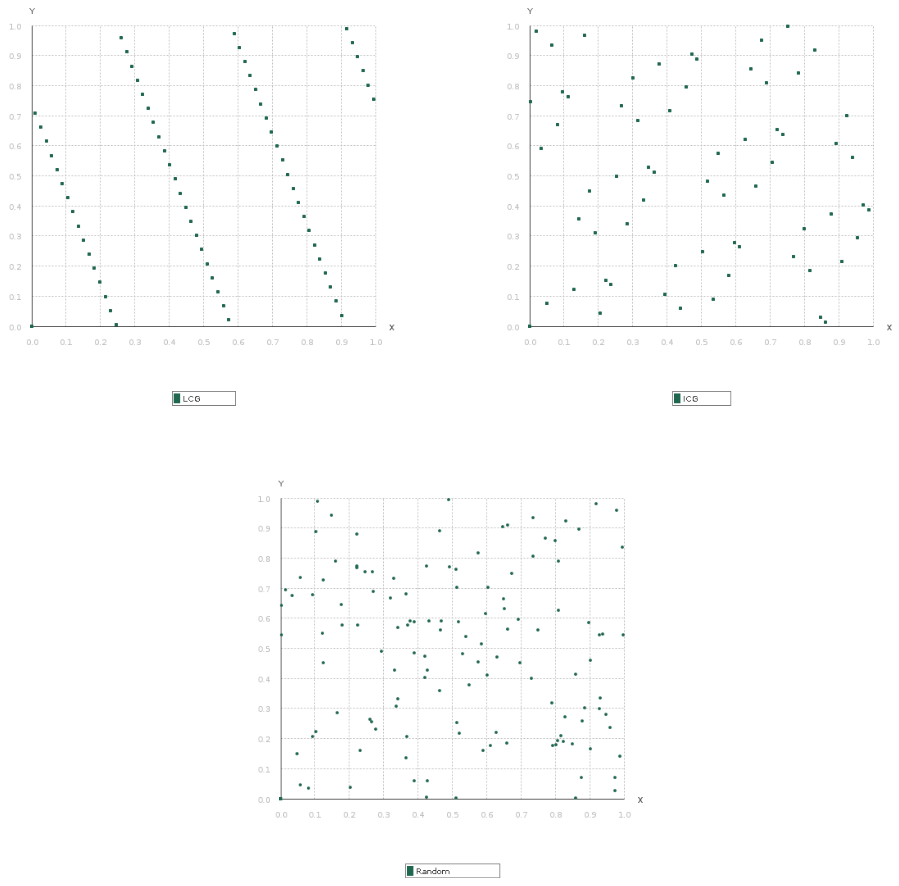

Figure 4.1: The output of an LCG (left) and an ICG (right), compared to true random numbers (bottom) from *random.org*.

[1]See *http://random.org/*

H_0	$0.00 \to 0.01$	$0.04 \to 0.05$	$0.08 \to 0.09$	$0.11 \to 0.12$
LCG	64	**4**	**4**	**4**
ICG	64	56	48	11
random.org	64	59	35	16
H_1	$0.00 \to 0.01$	$0.04 \to 0.05$	$0.08 \to 0.09$	$0.11 \to 0.12$
LCG	0	0	0	0
ICG	0	0	**6**	**6**
random.org	0	0	0	1

Table 4.1: Summary of the persistent homology of the “toy” generators.

We computed the persistent homology of these two generators using our algorithm. The results are summarized on Table 4.1. The parallel lines of the linear congruential generator are clearly visible in the zeroth homology: the point set has four connected components. Similarly, the round structures of the inversive generator show up in the first homology as six cycles.

4.3 Test Results

Next, we executed the following tests on a large set of generators to evaluate their performance.

- **UHHT**: Unit Hypercube Homology (H_0) with sample size $n = 12000$ in dimension 3.
- **MRHT**: Matrix Rank Homology (H_0) with sample size $n = 100$ in dimension 64.
- **SHT**: Subsequence Homology (H_1) with sample size $n = 50$ and bit length 64.

Each test was executed ten times, and every experiment was repeated twice. A p-value less than 0.01 was considered a failure. The detailed results can be found in the appendix. As expected, the RANDU generator systematically failed the UHHT test: similarly to the proof-of-concept, the existence of the fifteen parallel planes can easily be inferred from the test output. RANDU was the only generator to fail the SHT as well: in fact, this test proved quite unstable, possibly because of the low sample sizes.
All of the shift register generators systematically failed the MRHT test, along with Randu and the Glibc48 generator.

4.4 Conclusion

The proof-of-concept shows that the tests introduced in Chapter 3 can find topological structures hidden in pseudo-random sequences. The large-scale evaluation suggest that our new algorithms can reliably distinguish between high-quality and low-quality generators.
Comparing the results of the experiment and the proof-of-concept suggests that larger sample sizes could greatly increase the power of our tests. Unfortunately, the performance characteristics of the persistent homology algorithm make much larger samples intractable. Future work

should focus on dealing with this issue. Preliminary results suggest that taking a large sample and subsequently restricting the computations to a small subspace may be able to sidestep the problem.
While the test has not yet been able to uncover previously unknown failure modes in “real-world” generators, it has shown its versatility on the proof-of-concept instances. Once the performance issues are solved, the homology test will provide a useful - and extremely general - alternative to the spectral test.

Appendix: Test Results

UHHT

```
Randu                                  0/10
Randu                                  0/10
Minstd                                10/10
Minstd                                10/10
Glibc48                               10/10
Glibc48                               10/10
java.util.Random                      10/10
java.util.Random                      10/10
LFSR2547                              10/10
LFSR2547                              10/10
XorShift                              10/10
XorShift                              10/10
MersenneTwister                       10/10
MersenneTwister                       10/10
BlumBlumShub                          10/10
BlumBlumShub                          10/10
File-res-RandomOrg-Series1-50M        10/10
File-res-RandomOrg-Series2-50M        10/10
Quasirandom                            0/10

Took 732938ms
```

MRHT

```
Randu                            0/10
Randu                            0/10
Minstd                          10/10
Minstd                          10/10
Glibc48                          0/10
Glibc48                          0/10
java.util.Random                10/10
java.util.Random                10/10
LFSR2547                         0/10
LFSR2547                         0/10
XorShift                         0/10
XorShift                         0/10
MersenneTwister                 10/10
MersenneTwister                 10/10
BlumBlumShub                    10/10
BlumBlumShub                    10/10
File-res-RandomOrg-Series1-50M  10/10
File-res-RandomOrg-Series2-50M  10/10
Quasirandom                      0/10

Took 1417664ms
```

SHT

```
Randu                                 4/10
Randu                                 0/10
Minstd                               10/10
Minstd                               10/10
Glibc48                              10/10
Glibc48                               9/10
java.util.Random                      8/10
java.util.Random                     10/10
LFSR2547                              9/10
LFSR2547                             10/10
XorShift                             10/10
XorShift                              9/10
MersenneTwister                      10/10
MersenneTwister                       8/10
BlumBlumShub                          9/10
BlumBlumShub                         10/10
File-res-RandomOrg-Series1-50M        8/10
File-res-RandomOrg-Series2-50M       10/10
Quasirandom                           0/10

Took 121037ms
```

References

[1] Nina Amenta and Marshall Bern. Surface reconstruction by voronoi filtering. *14th Annual ACM Symposium on Computational Geometry, pages 39-48*, 1998.

[2] Michael Artin. *Algebra*. Prentice Hall, 1991.

[3] Lawrence E. Bassham, III, Andrew L. Rukhin, Juan Soto, James R. Nechvatal, Miles E. Smid, Elaine B. Barker, Stefan D. Leigh, Mark Levenson, Mark Vangel, David L. Banks, Nathanael Alan Heckert, James F. Dray, and San Vo. Sp 800-22-1a. A statistical test suite for Random and Pseudorandom Number Generators for cryptographic applications. Technical report, National Institute of Standards & Technology, Gaithersburg, MD, United States, 2010.

[4] Peter Bubenik and Jonathan A. Scott. Categorification of persistent homology. *Discrete and Computational Geometry*, 2014.

[5] David Cohen-Steiner, Herbert Edelsbrunner, and John Harer. Stability of persistence diagrams. *Discrete Comput. Geom.*, 2007.

[6] David Cohen-Steiner, Herbert Edelsbrunner, John Harer, and Konstantin Mischaikow. Extending persistence using poincare and lefschetz duality. *Foundations of Computational Mathematics*, 2009.

[7] International Business Machines Corporation. *System/360 Scientific Subroutine Package*. Version 3, 1968.

[8] M. D'amico and P. Frosini. Optimal matching between reduced size functions. Technical report, 2003.

[9] H. Edelsbrunner, D. Letscher, and A. Zomordian. Topological persistence and simplification. *Discrete Computational Geometry*, pages 39–48, 2002.

[10] Nicholas Enticknap. *Computing's Golden Jubilee*. The Computer Conservation Society, 2008.

[11] Patrizio Frosini. Stable comparison of multidimensional persistent homology groups with torsion. *Acta Appl. Math.*, pages 43–54, 2013.

[12] Robert Ghrist. Barcodes: The persistent topology of data. Technical report, 2007.

[13] F. James. Ranlux: A fortran implementation of the high-quality pseudorandom number generator of luscher. *Computer Physics Communications*, pages 111 – 114, 1994.

[14] Donald E. Knuth. *The Art of Computer Programming, Volume 2 (3rd Ed.): Seminumerical Algorithms*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1997.

[15] George Marsaglia. Random numbers fall mainly in the planes. Report, Boeing Scientific Research Laboratories, Seattle, WA, USA, August 1963.

[16] George Marsaglia. Xorshift RNGs. *Journal of Statistical Software*, 2003.

[17] W.S. Massey. *A Basic Course in Algebraic Topology*. Graduate Texts in Mathematics. Springer New York, 1991.

[18] Makoto Matsumoto and Takuji Nishimura. Mersenne twister: A 623-dimensionally equidistributed uniform pseudorandom number generator, 1998.

[19] S. K. Park and K. W. Miller. Random number generators: Good ones are hard to find. *Communications of the ACM*, October 1988.

[20] Carl E. Pierchala. An improvement for the mcgill university random number package. *Comput. Stat. Data Anal.*, 2(4):317–322, February 1985.

[21] Henri Poincare. Analysis situs. *Journal de l'ecole polytechnique*, pages 1–123, 1895.

[22] Andrey Sidorenko and Berry Schoenmakers. *Cryptography and Coding: 10th IMA International Conference, Cirencester, UK, December 19-21, 2005. Proceedings*, chapter Concrete Security of the Blum-Blum-Shub Pseudorandom Generator. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

[23] Andrew Tausz, Mikael Vejdemo-Johansson, and Henry Adams. JavaPlex: A research software package for persistent (co)homology. In Han Hong and Chee Yap, editors, *Proceedings of ICMS 2014*, Lecture Notes in Computer Science 8592, pages 129–136, 2014.

[24] United States. Navy Dept. Bureau of Ordnance. *Proceedings of a second symposium on large-scale digital calculating machinery*. Annals of the Computation Laboratory of Harvard University. Harvard University Press, 1951.

[25] John von Neumann. Various techniques used in connection with random digits. *J. Res. Nat. Bur. Stand.*, pages 36–38, 1951.