Exploring System Dynamics Using Topological Data Analysis

Md Abdul Gafur

Faculty of Health, Science and Technology

Master thesis in Mathematics

Second Cycle, 30 hp (ECTS)

Supervisor: Dr. Nikos Kavallaris, Karlstad University, Sweden

Co-advisor: Ole Sönnerborn, Karlstad University, Sweden

Examiner: Prof. dr habil. Adrian Muntean, Karlstad University, Sweden

Karlstad, May 30th, 2024

Abstract

The exploration of complex systems is a fundamental pursuit in various scientific disciplines, including physics, biology, finance and engineering. The inherent complexity and dynamics within these systems pose significant challenges for traditional analytical methods. In recent years, the emergence of Topological Data Analysis (TDA) has provided a promising framework for uncovering hidden structures and patterns in dynamic data sets.

This thesis investigates the application of Topological Data Analysis to analyze system dynamics, aiming to enhance our understanding of their behavior. Through a detailed review of existing literature, we examine the theoretical foundations of TDA and its relevance to discrete and continuous processes. We discuss conceptual underpinnings of persistent homology, a key technique in TDA, and its potential for capturing essential features of system dynamics. By applying TDA to two distinct models, the stochastic ODE and the discrete logistic equation, we demonstrate its effectiveness in revealing underlying structures that traditional methods might overlook, thereby offering new insights into the analysis of stochastic and discrete dynamical systems.

Keywords

Topological Data Analysis, Algebraic Topology, Simplicial Complex, Persistent Homology, Lazy Witness Filtration, Euler-Maruyama Scheme, Discrete Logistic Equation, Stochastic ODE.

MSC Classifications

$55N31, 62R40, 68T09$

Acknowledgments

I express my sincere gratitude to my esteemed supervisors, Dr. Nikos Kavallaris and Ole Sonnerborn, for their invaluable guidance and support throughout the duration of this thesis study. Their profound expertise and mentorship greatly contributed to the development of my understanding and knowledge in the field. Additionally, I extend my thanks to Dr. Konstantinos Spiliotis, Rostork University, Germany, for his assistance with MATLAB programming, which was instrumental in the execution of various aspects of this research.

I am also indebted to Professor dr. habil. Adrian Muntean for his instrumental role in providing me with an engaging and intellectually stimulating research topic, as well as for fostering a conducive mathematical environment throughout my academic journey at Karlstad University. Furthermore, I extend my appreciation to my program coordinator at the University of L'Aquila, Italy, specifically Professor Dr. Bruno Robino, for facilitating my transition to Karlstad University as part of the second phase of my degree and for ensuring a solid foundation of study during my time at L'Aquila.

Contents

Chapter 1

Introduction

1.1 Motivation

My primary motivation for undertaking this study stems from the burgeoning utilization of topological methodologies in the analysis of datasets, enabling the extraction of profound insights into their inherent shape and structure. Recent advancements have facilitated the seamless tracking of changes in the topological characteristics of these datasets, thereby enhancing our capacity to discern intricate patterns and relationships within them. It is this evolving landscape of topological data analysis that sparks intrigue, suggesting that such an approach could potentially furnish an alternative avenue for exploring the dynamics inherent in underlying dynamical systems. By harnessing the power of topological data methods, we aim to delve deeper into the dynamic interplay between various components of these systems, shedding light on their behavior and evolution over time with a novel perspective.

1.2 Main Objectives

This thesis investigates the dynamics of two models employing two distinct methodologies: classical and topological. Initially, we generate the dynamics using classical approaches. Subsequently, we apply topological methodologies to attain comparable dynamics. This dual-pronged approach facilitates a comparative analysis of both methods within the same context, aiming to shed light on their respective strengths and limitations in understanding system dynamics.

1.3 Outline

The thesis is structured as follows: Chapter 1 provides an overview of Topological Data Analysis (TDA). Chapter 2 delves into the methodology and provides a detailed discussion of all mathematical background relevant to this study. In chapter 3, we introduce two mathematical models upon which subsequent analyses are conducted. In Chapter 4, we present case studies illustrating the application of TDA in analyzing the dynamics. Finally, in chapter 5, we concludes with a summary of our findings and suggests future research directions like "Noise reduction analysis" in this field.

Chapter 2

Preliminaries

In this chapter, for readers convenience we present the main mathematical tools that we use throughout the work.

2.1 Elements of Stochastic Analysis

We start some necessary concepts of Stochastc Analysis.

2.1.1 Brownian Motion

Definition (Brownian motion): A stochastic process $w(t)$ for $t \in [0, T]$ is called *Brownian motion*, or *standard Wiener process*, over $[0, T]$ if it satisfies the following three conditions:

1. *Continuous Paths:* The *Brownian motion* has continuous paths almost surely. This means that, for any fixed t, the function $W(t)$ is continuous with probability 1. i.e., $W(0) = 0$.

2. *Stationary Increments:* For any $0 \leq s < t$, the increment $W_t - W_s$ is normally distributed with mean 0 and variance $t - s$: equivalently, $W_t - W_s \sim N(0, t - s)$.

3. *Independent Increments:* Increments of the Brownian motion over disjoint time intervals are independent. i.e., for $0 \leq s \leq t \leq u \leq v \leq T$, the increments $W(t) - W(s)$ and $W(v) - W(u)$ are independent.

2.1.2 Stochastic Differential Equation

From the deterministic perspective for an ordinary differential equation (ODE):

$$\frac{dx(t)}{dt} = f(x(t)), \tag{2.1}$$

we understand that a solution $x(t)$ is a function satisfying (2.1), given an initial condition $x(0)$. According to the Fundamental Theorem of Calculus, $x(t)$ can also be viewed as a function for which

$$x(t) - x(0) = \int_0^t f(x(s))\, ds.$$

This way of relating $x(t)$ to an integral equation can naturally extend to define a stochastic differential equation (SDE). Given functions f and g, we define the stochastic process $X(t)$ as a solution to the SDE:

$$dX(t) = f(X(t))\,dt + g(X(t))\,dW(t), \tag{2.2}$$

if $X(t)$ solves the integral equation

$$X(t) - X(0) = \int_0^t f(X(s))\,ds + \int_0^t g(X(s))\,dW(s). \tag{2.3}$$

Here, the second term on the right-hand side represents an Itô stochastic integral (as discussed in [4]), and the solution $X(t)$ is a random variable at each time t. Also the term $f(X(t))$ is called drift term while $g(X(t))$ is called the diffusion term of SDE (2.2). The symbols $dX(t)$, dt, and $dW(t)$ in (2.2) are primarily used for shorthand notation, indicating the integral formulation given in (2.3).

A commonly encountered SDE is the linear case with multiplicative noise, where the drift and diffusion terms are linear, that is $f(x) = \mu x$ and $g(x) = \sigma x$ in (2.2), leading to:

$$dX(t) = \mu X(t)\,dt + \sigma X(t)\,dW(t). \tag{2.4}$$

In 2.4, μ and σ are real constants. This SDE is frequently used to model the evolution of stock prices. If $\sigma = 0$ and $X(0)$ is a constant, then (2.4) simplifies to the ODE $dx(t) = \mu x(t)\,dt$, which has the solution $x(t) = x(0)e^{\mu t}$. This corresponds to a risk-free deposit in a bank with interest rate μ. The stochastic term $\sigma X(t)\,dW(t)$ represents random fluctuations, with σ known as the *volatility*. The solution to (2.4) is

$$X(t) = X(0)e^{(\mu - \frac{1}{2}\sigma^2)t + \sigma W(t)}. \tag{2.5}$$

Existence and uniqueness

It is pertinent to note that the mathematical theory concerning the existence and uniqueness of solution to SDEs typically imposes some conditions on the functions f and g in (2.2). The standard conditions underpinning much of this theory are that the drift and diffusion coefficients are globally Lipschitz continuous; that is, there exists a constant L such that f and g in (2.2) satisfy:

$$|f(X) - f(Y)| \leq L|X - Y| \text{ and } |g(X) - g(Y)| \leq L|X - Y| \quad \text{for all} \quad X, Y \in \mathbb{R}^{\times}.$$

We utilize these conditions when analyzing the convergence of a numerical method, while acknowledging that in many practical scenarios, these conditions may not be fully met.

2.1.3 Euler-Maruyama Scheme

The Euler-Maruyama method is a numerical technique for approximating solutions to stochastic differential equations (SDEs) that involve both deterministic and random components. These equations are often used to model systems subject to random fluctuations or noise. Consider the SDE:

$$dX(t) = f(X(t))\,dt + g(X(t))\,dW(t), \quad 0 \leq t \leq T, \quad \text{with } X(0) \text{ given.} \tag{2.6}$$

Using a standard discretization approach similar to that used for deterministic ordinary differential equations (ODEs), we define a step size $\Delta t = T/N$ for some integer N and compute approximate solutions at times $t_i = i\Delta t, i \in \mathbb{N}$. The exact solution of this SDE can be expressed as:

$$X(t_{n+1}) = X(t_n) + \int_{t_n}^{t_{n+1}} f(X(s))\,ds + \int_{t_n}^{t_{n+1}} g(X(s))\,dW(s). \tag{2.7}$$

Letting X_n denote our approximation to $X(t_n)$, the Euler-Maruyama method is defined as follows:

$$X_{n+1} = X_n + \Delta t f(X_n) + \Delta W_n g(X_n), \quad n \in \mathbb{N} \tag{2.8}$$

where $\Delta W_n := W(t_{n+1}) - W(t_n)$ is a random variable sampled from a standard normal distribution ($\Delta W_n \sim N(0, \Delta t)$). This term introduces the randomness associated with the Brownian motion (as discussed in Section 2.1.1). For more details on section 2.1, one can see at [4].

2.2 Discrete Difference Equations (DDE)

Definition (Difference Equation): A first-order difference equation is an equation:

$$x_t = f(t, x_{t-1}),$$

where f is a function of two variables. A solution of the first-order difference equation $x_t = f(t, x_{t-1})$ is a function x of a single variable whose domain is the set of integers such that $x_t = f(t, x_{t-1})$ for every integer t, where x_t denotes the value of x at t.

We can find a solution of a first-order difference equation by successive calculation. That is, given the value of x for value of t_i, we can use the equation to find the value of x at the next value of t_{i+1}, and then use the equation again to find the value of x at the following value of t, and so forth. For example, given the value x_0 of x at 0, we have

$$x_1 = f(1, x_0)$$
$$x_2 = f(2, x_1) = f(2, f(1, x_0))$$

and so on.

In fact, this argument establishes that every first-order difference equation has a unique (as f is a deterministic function and the initial condition x_0 is uniquely specified) solution: given any value of x_0, there exists a unique solution x, with values $x_1, x_2, \ldots$.

Proposition 2.2.1. *For every number x_0, every first-order difference equation $x_t = f(t, x_{t-1})$ has a unique solution in which the value of x is x_0 at 0.*
Proof: See [6].

2.2.1 Periodic Behaviour

- In a difference equation, periodic behavior occurs when the sequence generated by the equation repeats itself after a certain number of steps.
- Mathematically, let us say we have a difference equation $x_{n+1} = f(x_n)$ where x_n represents the value of the sequence at time n.
- If there exists $T \in \mathbb{N}$ such that $x_{n+T} = x_n$ for all $n \in \mathbb{N}$, then the sequence is periodic with period T.

- Periodic behavior often arises when the difference equation exhibits stable, oscillatory dynamics, causing the sequence to repeat the same pattern over and over again.

For a more detailed description on periodic behaviour the reader may consult [6].

2.2.2 Chaotic Behaviour

Chaotic behavior in the discrete logistic equation refers to the sensitive dependence on initial conditions and the complex, unpredictable dynamics that can arise in the population model. It is often noticed by divergent paths in parameter space which seem to change randomly, but which actually result from some kind of time-dependent deterministic process.

- Chaotic behavior, on the other hand, arises when the sequence generated by the difference equation exhibits extreme sensitivity to initial conditions.
- Mathematically, this sensitivity means that small changes in initial conditions lead to drastically different outcomes over time.
- Chaotic behavior typically arises in the nonlinear dynamics of some difference equations, causing the system to exhibit complex, irregular behavior that appears unpredictable.
- Despite its apparent randomness, chaotic behavior in a deterministic system is governed by underlying mathematical principles and is not truly random.

For a more detailed description of chaotic behaviours related to difference equations, the reader is refer to [6].

2.2.3 Examples of Difference Equations

Population Dynamics

Logistic Growth Model: Difference equations can be used to model population growth where the population size P at time $t+1$ depends on the population size at time t:

$$P_{t+1} = P_t + rP_t\left(1 - \frac{P_t}{K}\right)$$

where r is the growth rate and K is the carrying capacity of the environment. This example is taken from [6].

Economics

Cobweb Model: This model describes the price dynamics of a market where the price P at time t depends on the price and quantity at previous times:

$$P_{t+1} = a - bQ_t$$

$$Q_t = c + dP_t$$

where a, b, c, and d are constants. This model is used to analyze supply and demand interactions over time.This example is taken from [6].

Finance

Compound Interest: The future value of an investment based on compound interest can be modeled using a difference equation:

$$A_{t+1} = A_t(1 + r)$$

where A_t is the amount of money at time t and r is the interest rate. This example is taken from [6].

2.3 Elements of Algebraic Topology

In the final stage of our work is on topological data analysis (TDA) with a focus on algebraic topology. So, it is important to cover the following key topics: Simplicial complex, simplicial homology, Betti number and persistent homology. We are taking all the therioes and the related description from [1].

2.3.1 Simplicial Complex (Unoriented)

[*Unoriented* $\sim$ *in our case we will do homology with* $\mathbb{Z}_2$*-coefficients*]

Definition(Simplicial Complex): If $V = \{v_0, v_1, ..., v_n\}$ be a set of finite numbers of vertices. Then, a simplicial complex K with vertex set V is a set of subsets of V satisfying:

1. $V \subseteq K$ i.e. $\forall\{v_j\} \in K$
2. If $\sigma = \{v_{i_0}, v_{i_1}, ..., v_{i_m}\} \in K$, thus every subset $\tau \subseteq \sigma$ belongs to K.

In our case, the data points are referring the vertices. Given a simplicial complex, we can "cook up" a

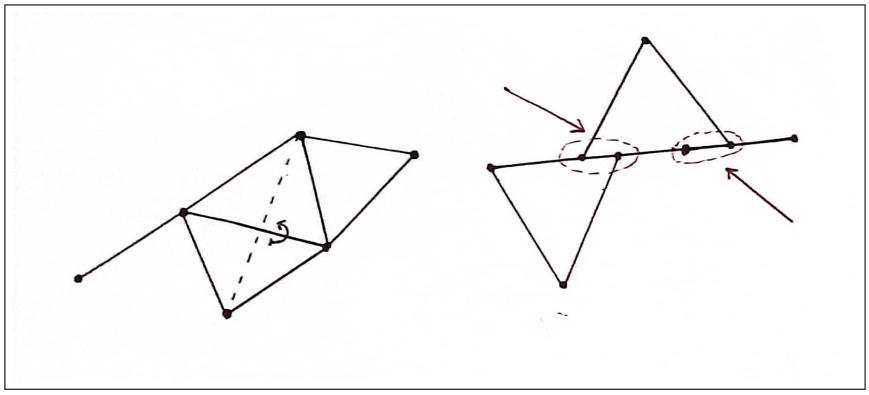

Figure 2.1: Left: Example of a simplicial complex. Right: Union of simplices which is not a simplicial complex.

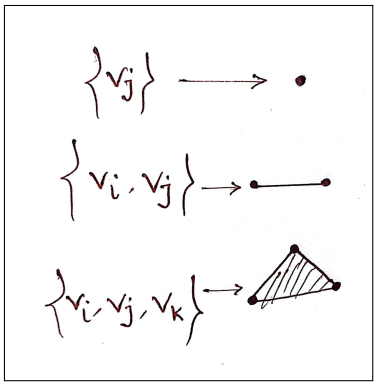

Figure 2.2: Basic simplices.

homological algebra: Elements of K are called simplicies. Specifically, if σ contains $k + 1$ elements, σ is called a k-th simplex. For each k, $C_k(K)$ is the $\mathbb{Z}_2$-vector space with the basis of k-simplicies in K:

$$C_k(K) = \{\sum_{j=1}^{N} n_i\sigma_i : \sigma_j \quad k - simplex, n_j \in \mathbb{Z}_2\}$$

Boundary Homomorphism (Linear Maps)

Definition (Boundary Homomorphism): We define the boundary homomorphisms of k-simplex as follows:

$$\partial_k : C_k(K) \to C_{k-1}(K)$$
$$\partial_k \left(\sum n_j \sigma_j \right) = \sum n_j \partial_k(\sigma_j)$$

If $\sigma_j = \{v_{i1}, v_{i2}, ..., v_{ik+1}\}$, then the boundary of σ_j is:

$$\partial_k (\sigma_j) = \sum_{l=1}^{k+1} \{v_{i1}, v_{i2}, ..., \hat{v}_{il}, ..., v_{ik+1}\},$$

where $\hat{v}_{il}$ means that the vertex v_{il} is omitted.

Definition (Chain Complex): The collection of sequences of vector spaces $C_k(K)$ indexed by integers k is called the chain complex.

Each group $C_k(K)$ represents the vector space generated by k-dimensional simplices in the topological space. For example, $C_0(K)$ consists of the vector space generated by vertices, $C_1(K)$ by edges, $C_2(K)$ by triangles, and so on.

$$\xrightarrow{\partial_{k+1}} C_k(K) \xrightarrow{\partial_k} C_{k-1}(K) \xrightarrow{\partial_{k-1}} \cdots \xrightarrow{\partial_2} C_1(K) \xrightarrow{\partial_1} C_0(K) \xrightarrow{\partial_0} 0$$

where 0 represents the trivial group.

Proposition 2.3.1. *The boundary of the boundary of a chain is always zero:*

$$\partial_{k-1} \circ \partial_k = 0.$$

Proof: *See [1].*

Definition (K-cycles): An element c in the kernel of ∂_k is called a *k-cycles.* For $k \in \{0, \ldots, d\}$, the set $Z_k(K)$ of k-cycles of K is the kernel of $\partial_k : C_k \to C_{k-1}$:

$$Z_k(K) := \ker(\partial_k) = \{c \in C_k(K) : \partial_k c = 0\}.$$

Definition (K-boundary): An element c in the image of ∂_{k+1} is called a k-boundary. The image $B_k(K)$ of $\partial_k : C_{k+1}(K) \to C_k(K)$ is the set of k-chains bounding a $(k+1)$-chain:

$$B_k(K) := \mathrm{im}(\partial_k) = \{c \in C_k(K) : \exists c_0 \in C_{k+1}, c = \partial c_0\}.$$

$B_k(K)$ and $Z_k(K)$ are subgroups of $C_k(K)$ and according to the Proposition 2.4.1, one has

$$B_k(K) \subseteq Z_k(K) \subseteq C_k(K).$$

Figure 2.3 presents the examples of chains, cycles and boundaries

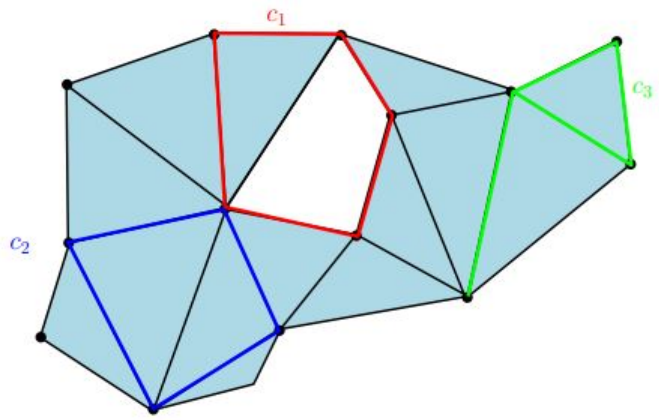

Figure 2.3: Examples of chains, cycles and boundaries: c_1 is a cycle which is not a boundary, c_2 is a boundary and c_3 is a chain that is not a cycle. Source: [1]

Definition (Homology Group): The homology group $H_k(K)$ of dimension k is defined as the quotient group of cycles modulo boundaries:

$$H_k(K) = \frac{Z_k(K)}{B_k(K)} = \frac{\ker(\partial_k)}{\text{im}(\partial_{k+1})}$$

Homology group quantify the "holes" or "voids" in the space represented by the simplicial complex. Intuitively, they capture different-dimensional loops, voids, and connected components in the space. The k^{th} homology group, denoted as H_k, represents k-dimensional cycles (cycles that do not bound any (k+1)-dimensional region) modulo k-dimensional boundaries.

2.3.2 Persistent Homology

We recall the two axioms behind the simplicial complex:

1. $V \subseteq K$ i.e. $\forall\{v_j\} \in K$
2. If $\sigma = v_{i_0}, v_{i_1}, ..., v_{i_m} \in K$, thus every subset $\tau \subseteq \sigma$ belongs to K.

Čech and Vietoris-Rips Filtrations

If V is part of a metric space, then we can create simplicial complex in the following way:

Definition (Čech Filtration): Given $r > 0$, the Čech complex with vertex set V and parameter r is the nerve $\check{\text{C}}\text{ech}(V, r)$ of the unions of balls centered on V with radius r. The simplices of $\check{\text{C}}\text{ech}(V, r)$ are characterized by the following condition:

$$\{v_{i_0}, v_{i_1}, ..., v_{i_m}\} \in \check{\text{C}}\text{ech}(V, r) \iff \bigcap_{j=1}^{k+1} B_r(v_{ij}) \neq \emptyset.$$

Definition (Vietoris-Rips Filtration): Given $r > 0$, the Vietoris-Rips complex $\text{Rips}(V, r)$ with vertex set V and parameter r is defined by the following condition:

$$\{v_{i_0}, v_{i_1}, ..., v_{i_m}\} \in \text{Rips}(V, r) \iff \|x_{ii} - x_{ij}\| \leq r \text{ for all } j \in \{0, \ldots, k\}.$$

Proposition 2.3.2. *For any $r \geq 0$,*

$$Rips(V, r) \subseteq \check{C}ech(V, r) \subseteq Rips(V, 2r)$$

Proof*: See [1].*

Betti Number

The *Betti numbers* are topological invariants that describe the connectivity and holes in a topological space. They help characterize the topology and shape of spaces, classify geometric objects, and distinguish between different types of spaces based on their connectivity and holes.

Definition (Betti Number): Given a simplicial complex K, the k^{th} Betti number, denoted as $\beta_k(K)$, is the rank of the k^{th} homology group $H_k(K)$. Mathematically:

$$\beta_k(K) = \text{rank}(H_k(K))$$

Betti numbers provide information about the number of k-dimensional holes in X.

- For $k = 0$, $\beta_0(X)$ counts the number of connected components.
- For $k = 1$, $\beta_1(X)$ counts the number of "1-dimensional holes" or "loops".
- Higher Betti numbers ($k > 1$) represent higher-dimensional holes.

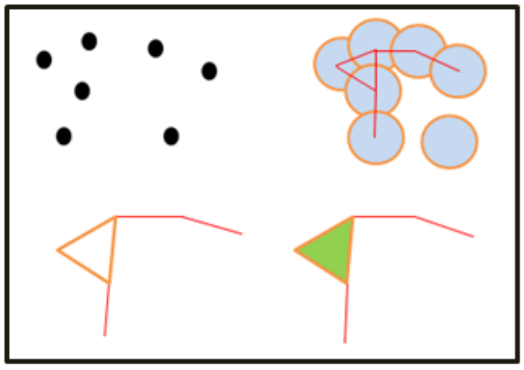

Figure 2.4: $\beta_0 = 2, \beta_1 = 1, \beta_2 = 0$.

An important observation is that Čech and Vietoris-Rips filtrations added a dynamical parameter r. For every $r \geq 0$ one can get a simplicial complex $K(r)$. That means one can get new $Z_k(K(r))$, $B_k(K(r))$ and $C_k(K(r)$ depending on r! Also,

$$Z_k(K(r_1)) \subseteq Z_k(K(r_2)) \quad if \quad r_1 \leq r_2$$

Computation of Betti Numbers

Consider a finite simplicial complex K of dimension d with a filtration

$$\emptyset = K^0 \subset K^1 \subset \cdots \subset K^m = K,$$

where for each $i = 0, \ldots, m-1$, the complex K^{i+1} is obtained by adding a simplex σ^{i+1} to K^i:

$$K^{i+1} = K^i \cup \sigma^{i+1}.$$

Assume that the Betti numbers for K^{i-1} have already been determined. When adding the simplex σ^i of dimension $k+1$ to form K^i, note that by the filtration's construction, σ_i cannot be part of the boundary of any $(k+2)$-simplex in K^i. Consequently, if σ^i is included in a $(k+1)$-cycle in K^i, this cycle is not the boundary of any $(k+2)$-chain in K^i.

Definition (Positive-Negative): Let K be a d-dimensional simplicial complex and let

$$\emptyset = K^0 \subset K^1 \subset \cdots \subset K^m = K,$$

be a filtration of K. A simplex σ^i is called positive if it is contained in a $(k+1)$-cycle in K^i (which is necessarily not a boundary in K^i) and negative otherwise.

With the aforementioned definition, the k-th Betti number of K is equal to the difference between the number of positive k-simplices (which are creating k-cycles) and the number of negative $(k+1)$-simplices (which are "killing" k-cycles).

Persistent Betti Number

Let K be a d-dimensional simplicial complex and let

$$\emptyset = K^0 \subset K^1 \subset \cdots \subset K^m = K,$$

be a filtration of K such that for any $i = 0, \ldots, m-1$,

$$K^{i+1} = K^i \cup \sigma^{i+1},$$

where σ^{i+1} is a simplex.
For any $0 \leq n \leq m$, we denote by C_k^n the set of k-chains (with coefficients in $\mathbb{Z}/2\mathbb{Z}$) of K^n. Notice that the restriction of the boundary operator to C_k^n has its image contained in C_{k-1}^{n-1}. We denote by Z_k^n and B_k^n the sets of k-cycles and k-boundaries of K^n respectively. The k-th homology group of K^n is thus

$$H_k^n = \frac{Z_k^n}{B_k^n}.$$

With these notations, we have the following inclusions:

$$Z_k^0 \subset Z_k^1 \subset \cdots \subset Z_k^n \subset \cdots \subset Z_k^m = Z_k(K)$$

$$B_k^0 \subset B_k^1 \subset \cdots \subset B_k^n \subset \cdots \subset B_k^m = B_k(K)$$

Definition (Persistent Betti Number): For $p \in \{0, \ldots, m\}$ and $l \in \{0, \ldots, m-p\}$, the k-th persistent Betti number of K_l is the dimension of the vector space

$$H_k^{l,p} = \frac{Z_k^l}{B_k^{l+p} \cap Z_k^l}.$$

2.3.3 Software (Javaplex[7])

Javaplex is a software package designed for computational topology and topological data analysis (TDA). It provides tools for analyzing and visualizing topological features of data sets, particularly focusing on persistent homology, a method used to extract topological signatures from data.

Javaplex emerged from the need for a flexible platform to support new research directions in topological data analysis and computational persistent homology. The website for Javaplex is `http://appliedtopology.github.io/javaplex/`, the documentation overview is available at `https://github.com/appliedtopology/javaplex/wiki/Overview`, and the javadoc tree for the library can be found at `http://appliedtopology.github.io/javaplex/doc/`.

For more details, we refer the reader to the section A.9 in appendix.

Chapter 3

The Mathematical Models

In this section, we present two models: one based on stochastic processes and the other on discrete systems.

3.1 A Stochastic Model with Double-well Potential

Stochastic differential equations (SDEs) are a class of mathematical models used to describe systems that are influenced by random forces. One particularly interesting and widely studied example of SDEs is those with a drift term arising from a double-well potential.

A double well potential is a type of potential energy function characterized by having two local minima separated by a barrier. This potential is a prototypical model for studying systems that exhibit bi-stability, where the system can exist in one of two states corresponding to the minima of the potential. In physical terms, these states can represent, for example, the magnetization direction in a ferromagnetic material, the conformation of a bio molecule, or the phase of a substance undergoing a phase transition.

When considering the dynamics of a particle in a double well potential under the influence of random noise, the system can be modeled by a stochastic differential equation of the form:

$$dX(t) = -\nabla V(X(t))dt + \sigma dW_t$$

where $X(t)$ denotes the position of the particle at time t, $V(X)$ is the double well potential, σ is the noise intensity, and $W(t)$ represents a Wiener process or Brownian motion. The term $-\nabla V(X(t)$ is the drift term that drives the particle towards the minima of the potential, while the term $\sigma dW(t)$ represents the stochastic noise that perturbs the system, see [3, 9].

The double well potential is a form such as:

$$V(x) = \gamma\frac{x^4}{4} - \mu\frac{x^2}{2}$$

where μ and γ are positive constants. For simplicity, in the following we take $\gamma = 1$. This potential has two minima at $x = \pm\sqrt{\mu}$ and a maximum at $x = 0$, see Figure 3.1. The dynamics governed by the SDE encapsulate the competition between the deterministic force driving the system towards these minima and the random fluctuations pushing the system away from them. This interplay leads to rich and complex behavior, including the possibility of the particle transitioning from one well to the other, a

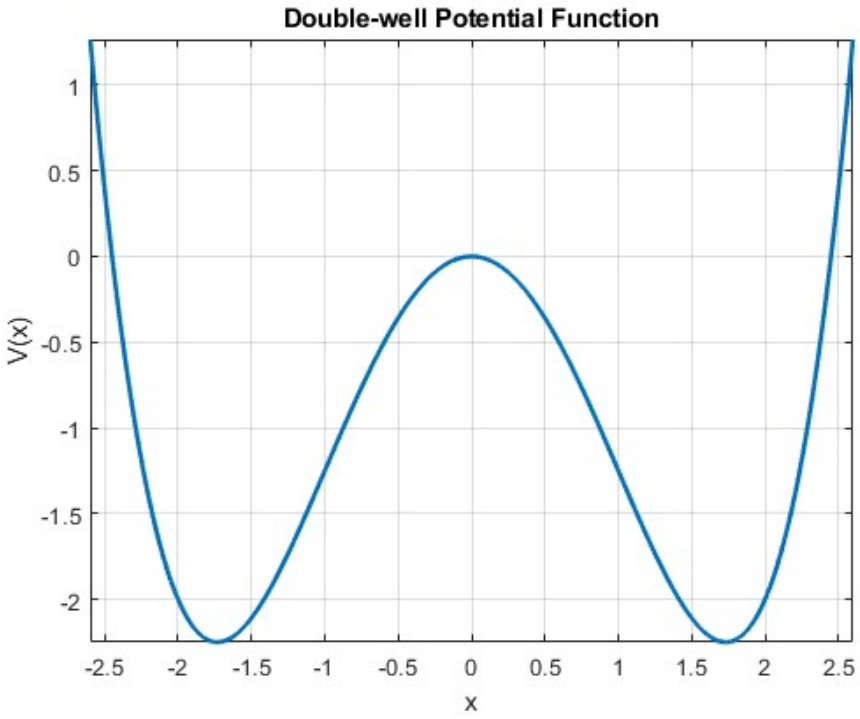

Figure 3.1: A graphical representation, of double-well, potential non-linearity.

phenomenon known as noise-induced transitions.

Analyzing SDEs with a drift term from a double well potential provides insights into the stability of systems under random perturbations, the rates of transitions between states, and the influence of noise on system behavior. These insights are crucial in understanding natural phenomena and in designing systems that either leverage or mitigate the effects of randomness. Whether modeling the switching of electronic components in computing devices, the conformational changes of biomolecules, or the behavior of financial markets, SDEs with a double well potential offer a powerful framework for studying and predicting the dynamics of complex systems under uncertainty, see [2].

In the first model we explore its dynamics using persistent homology is the following stochastic model:

$$dX_t = (\mu X_t - X_t^3)dt + \sigma dW_t; \quad t > 0, \delta > 0, \tag{3.1}$$

where δ is the position of the particle at time $t = 0$, μ determines the depth of the vallies and σ is the strength of the source of randomness.

3.2 A Discrete Logistic Equation Model

The discrete logistic equation is a fundamental model in population dynamics and mathematical biology, offering profound insights into the behavior of populations over time. Originally formulated by the Belgian mathematician Pierre François Verhulst in the 19^{th} century, see [10], and later popularized by Robert May in the 20^{th} century, see [5], this equation provides a simple yet powerful framework for understanding how populations grow and stabilize under various constraints.

At its core, the discrete logistic equation is given by:

$$x_{t+1} = \alpha x_t \left(1 - \frac{x_t}{K}\right), \quad t > 0, \tag{3.2}$$

where:

- x_t represents the population size at the t^{th} time step,
- α is the intrinsic growth rate of the population,
- K is the carrying capacity of the environment, representing the maximum population size that the environment can sustain.

This equation encapsulates two critical forces in population dynamics: reproduction and density-dependent regulation. The term αx_t models exponential growth, indicating that the population can potentially grow without bounds in the absence of limiting factors. However, the term $\left(1 - \frac{x_t}{K}\right)$ introduces a feedback mechanism that reduces the growth rate as the population size approaches the carrying capacity K. This negative feedback is essential for modeling realistic population dynamics, where resources such as food, space, and other necessities are limited. For simplicity in the following we only consider the case $K = 1$ and thus we will examine the following discrete model:

$$\begin{cases} x_{t+1} = \alpha x_t \left(1 - x_t\right), \quad t > 0, \\ x_0 = \beta > 0, \end{cases} \tag{3.3}$$

where β stands for the initial condition representing the initial population concentration. The discrete logistic equation is not only mathematically elegant but also exhibits a rich variety of dynamical behaviors. For small values of α, the population tends to a stable equilibrium at 1 reflecting a balance between birth and death rates. As α increases, the system can exhibit periodic oscillations, with the population fluctuating regularly around the carrying capacity. For even larger values of α, the system can enter a regime of chaos, where the population size changes unpredictably from one generation to the next. This sensitivity to initial conditions and parameters makes the logistic equation a cornerstone for studying complex dynamical systems.

Beyond its theoretical significance, the discrete logistic equation has practical applications in ecology, conservation biology, and resource management. It helps ecologists predict population trends, assess the impact of environmental changes. Moreover, the insights gained from this model extend to other fields, such as epidemiology, where similar principles govern the spread of diseases, and economics, where the growth of markets can be studied using analogous equations.

In summary, the discrete logistic equation serves as an important tool for understanding and predicting the dynamics of populations. Its ability to model growth, stability, and chaos within a simple mathematical framework underscores its enduring relevance and utility across diverse scientific disciplines.

Chapter 4

Main Results

Our work follows three main steps to make sure we look at everything closely and compare it well. First, we start with what we call the "Classical Approach" (the Euler-Maruyama method) using proper discretizations of the underlying models. Proper discretization involves converting continuous models or equations into discrete forms that can be handled computationally. This process ensures that the discretized model accurately represents the original continuous model, preserving its essential properties and behavior. This is where we carefully put together all the data we need for our models.

Next, we move on to using persistent homology from Topological Data Analysis to produce an alternative data set (time series). These tools help us see the complex structures and connections in our data. By using them, we uncover hidden patterns and details that help us understand things better.

Finally, we bring together the results from both approaches and compare them closely. We carefully look at what each approach shows us, what it does well, and where it might have limitations. This last step helps us combine theory and real-world observations, so we can make smart decisions and improve our models even more.

4.1 A SDE Model with Double-well Potential Drift Term

4.1.1 Classical Approach: Euler-Maruyama Scheme

Here we use the Euler-Maruyama approximation scheme to solve this stochastic model numerically:

$$\begin{cases} dX(t) = (\mu X(t) - X^3(t))dt + \sigma dW_t, \quad t > 0, \\ x_0 = \delta > 0, \end{cases} \tag{4.1}$$

The Euler-Maruyama approximation for this SDE is given by:

$$X_{n+1} = X_n + (\mu X_n - X_n^3)\Delta t + \sigma \Delta W_n,$$

where Δt is the time step size and ΔW_n is a sample from a normal distribution with mean 0 and variance Δt.

Given $\mu = 1.0$, $\sigma = 0.5$, $\delta = 1.0$, $T = 1.0$, and $N = 1000$ time steps, we'll compute the solution numerically. Then we produce data in figure 4.1 by MATLAB simulation (see A.1).

Now, the idea is to normalise out the time series $X(t) \in [-1, 1]$. To this end we consider

$$d(t) = \frac{X(t) - X_{min}}{X_{max} - X_{min}} \in [0, 1],$$

where X_{min}, X_{max} represent the minimum and maximum values of $X(t)$ over the considered time interval. We then generate the normalised version of our data in figure 4.2.

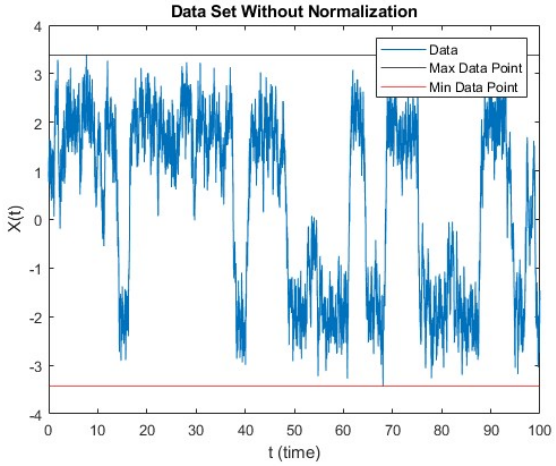

Figure 4.1: Here is the data generated relative to the time indicated on the $x-$axis. Instability is observed around t=0.5, and we can observe bi-stability in the fluctuations both above and below this point. For code, see A.1.

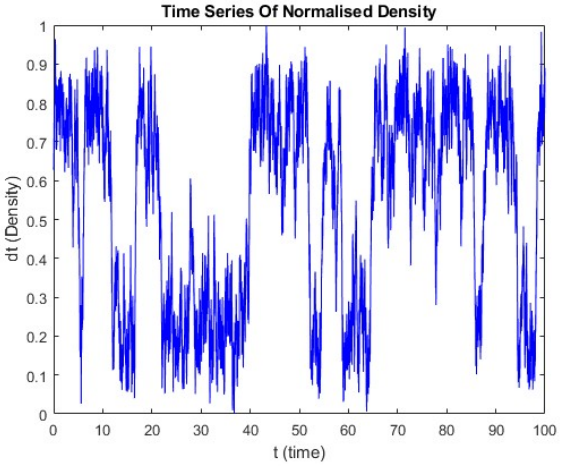

Figure 4.2: Normalized version of $X(t)$ from fig 4.1.

Given that the normalized time dataset $d(t)$ lies within the interval $[0, 1]$, we leverage this normalization at a later stage to map these values onto the unit circle. Subsequently, we employ a topological approach to generate a corresponding relative dataset.

4.1.2 Topological Approach: Persistent Homology

Next we choose $N = 10000$ equidistant points on the unit circle. The code calculates the coordinates of these points using the parametric equations of a circle:

$$x_j = \cos(\theta_j)$$

$$y_j = \sin(\theta_j)$$

where $\theta_j = 0 : \frac{2\pi}{N} : 2\pi \Longrightarrow \theta_j = \frac{2\pi_j}{N}; j = 0, 1, 2, \ldots, N-1$. See Figure 4.3.

Initially, we consider all of the points ($N = 10000$) are inactive. Then, for each time step t_j, we randomly active $[d(t_j) \cdot N]$ data points, see Figure 4.4. Then, we start growing circles around those

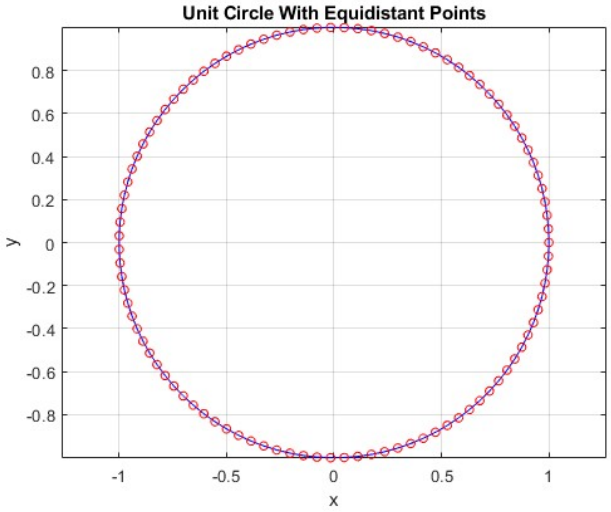

Figure 4.3: Set of inactive points:
$\mathbb{X}_I = (x_1, y_1; x_2, y_2; .....; x_N, y_N)$

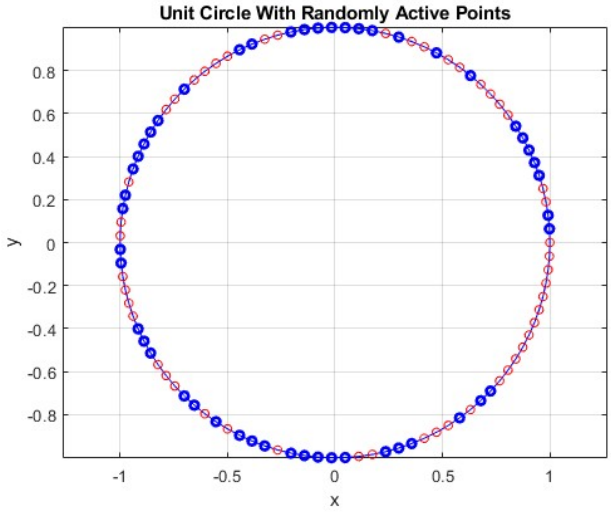

Figure 4.4: Set of activated points:
$\mathbb{X}_A = (x_{i_0}, y_{i_0}; x_{i_1}, y_{i_1}; .....; x_{i_{N-1}}, y_{i_{N-1}})$

Figure 4.5: Left: Sample of a unit circle with inactive points. Right: Sample of a unit circle with randomly active points. In those picture, we considered N=100 points only to get a better visualization. But, in the final coding we will take N=10000. (For code see A.2).

active nodes until the $Betti_1$ is born and store the radius $r(t_j)$ when this occurs. For more details on this process, one can see at section A.9 where I described how javaplex works with different functions. This process is repeated for different time steps $t_1, t_2, \ldots, t_n$ and then we consider the minimum radius $r_{min}(t) = \{r(t_1), r(t_2), \ldots, r(t_n)\}$.

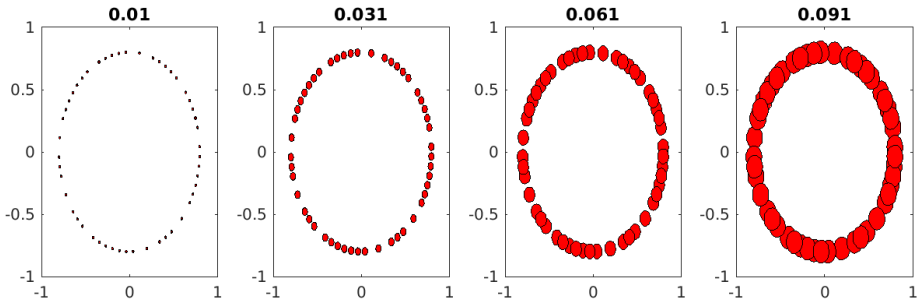

Figure 4.6: The figure illustrates the filtration process in topological data analysis using a series of disks with radius r. Each disk is centered at the given points in the dataset. As r increases, the disks grow, and edges are drawn between points that are within distance r of each other. This process helps visualize the birth of Betti numbers: Betti-0 (connected components) and Betti-1 (loops). The plot updates dynamically with each increase in r, showing how topological features emerge and evolve. The current value of r is displayed in the title. This is a sample of filtration process that how the $Betti_1$ is born. We use Vietoris-Rips Filtration (2.3.2) in Javaplex (A.9.2). [For code see A.3].

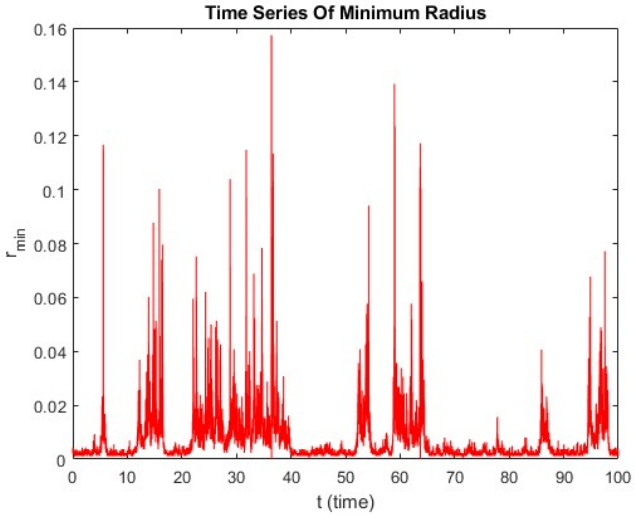

Figure 4.7: New time series with the collection of all r_{min} over n time steps [For code see at A.4].

4.1.3 Comparison and Results

Visual Distribution

We now have two data sets, one from the Euler-Maruyama method and another from the topological filtration process. We can compare both the datasets by plotting them in one Figure 4.8.

Observations

In Figure 4.8, we present the dynamics of two distinct time series: $d(t)$ and $r_{\min}(t)$. It is evident that as the density time series $d(t)$ increases, the minimum radius time series $r_{\min}(t)$ decreases.

The plot clearly demonstrates that periods of high density of time series $d(t)$ correspond to shorter minimum radii $r_{\min}(t)$, indicating a pronounced inverse relationship between these two variables. This phenomenon can be explained by the fact that higher density signifies a greater concentration of data

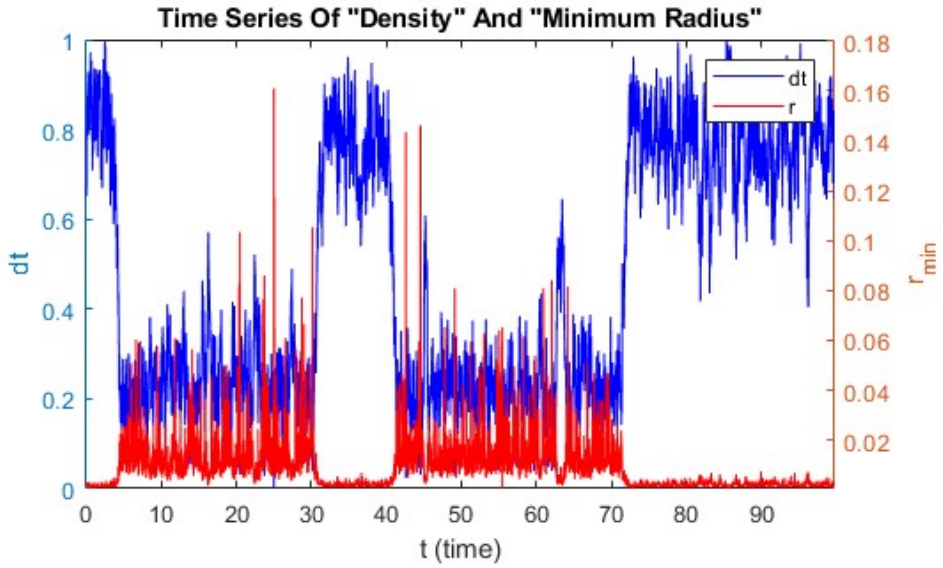

Figure 4.8: Here, the blue curve corresponds to the real dynamics while the red one is the minimal radius for the formation of a circle ($Betti_1 = 1$), in the filtration process. We use $d(t)$ instead of $X(t)$ to refer the density of data set. [For code one can see at A.4]

points within a given area, thereby reducing the distance required to connect points and form a topological feature, such as a circle, during the filtration process. Conversely, when the density is lower, data points are more sparsely distributed, necessitating a larger radius to achieve the same topological connections.

This observation highlights the sensitivity of topological features to the underlying density of the dataset, maintaining the same dynamics as the classical approach. It indicates that topological methods are effective in capturing and reflecting changes in data density. This capability can be valuable for various applications, such as identifying clustering tendencies or detecting shifts in data distribution over time.

4.2 Discrete Logistic Equation

4.2.1 Classical Approach: Numerical Simulation

Stability Analysis

Let us now explore one important characteristic of the discrete logistic model: its equilibria. As discussed previously in Section 3.2, this difference equation exhibits diverse behaviors, including periodic and chaotic dynamics, as the value of α increases. However, in this thesis, we will only focus exclusively on the periodic behavior. To examine the equilibria of the following discrete logistic model:

$$\begin{cases} x_{t+1} = \alpha x_t \left(1 - x_t\right), & t > 0, \\ x_0 = \beta > 0, \end{cases} \tag{4.2}$$

where β represents the initial population concentration. To find the equilibria, we set $x_{t+1} = x_t = x^*$, where x^* is a fixed point. Substituting this into the equation 4.2 gives:

$$x^* = \alpha x^*(1 - x^*)$$

$$x^*(\alpha - \alpha x^* - 1) = 0$$

This equation is satisfied if either $x^* = 0$ or $\alpha - \alpha x^* - 1 = 0$.

1. Equilibrium at $x^* = 0$:
$$x^* = 0$$

2. Non-trivial Equilibrium: Solving $\alpha - \alpha x^* - 1 = 0$:
$$\alpha(1 - x^*) = 1$$
$$x^* = 1 - \frac{1}{\alpha}$$

Finally, the discrete logistic model has two equilibria: Trivial Equilibrium: $x^* = 0$ and non-trivial Equilibrium: $x^* = 1 - \frac{1}{\alpha}$, provided $\alpha > 1$.

To determine the stability of these equilibria, we can analyze the derivative of the function $f(x) = \alpha x(1 - x)$ at the equilibrium points. The derivative of $f(x)$ is:

$$f'(x) = \alpha - 2\alpha x$$

The stability condition requires $|f'(x^*)| < 1$.

- At $x^* = 0$ we have
$$f'(0) = \alpha$$
  - If $|\alpha| < 1$, the equilibrium at $x^* = 0$ is stable.
  - If $|\alpha| > 1$, the equilibrium at $x^* = 0$ is unstable.
- At $x^* = 1 - \frac{1}{\alpha}$ we have:
$$f'\left(1 - \frac{1}{\alpha}\right) = \alpha - 2\alpha\left(1 - \frac{1}{\alpha}\right) = \alpha - 2\alpha + \frac{2\alpha}{\alpha} = \alpha - 2\alpha + 2 = 2 - \alpha$$

- If $|2 - \alpha| < 1$, the equilibrium at $x^* = 1 - \frac{1}{\alpha}$ is stable. - This simplifies to $1 < \alpha < 3$ for stability.

The discrete logistic model has the following equilibria and stability characteristics:

- Equilibrium at $x^* = 0$: - Stable if $0 < \alpha < 1$ - Unstable if $\alpha > 1$
- Equilibrium at $x^* = 1 - \frac{1}{\alpha}$: - Exists for $\alpha > 1$ - Stable if $1 < \alpha < 3$ - Unstable if $\alpha > 3$

The above analysis provides insight into how the population concentration evolves over time based on the parameter α.

First, let us systematically examine all the behaviors exhibited by this difference equation. To generate the time series of the behaviors, we utilize two separate MATLAB files. One file will describe the logistic function, while the other will handle the iteration process.

First we plot the bifurcation diagram of the long-term behavior for the logistic map generated by iterating over a range of values for parameter α. The logistic map equation function computes the time series for each α, and the main script stores and plots these values to visualize the bifurcation structure.

Then, we plot the behaviours for specific values of α as follows:

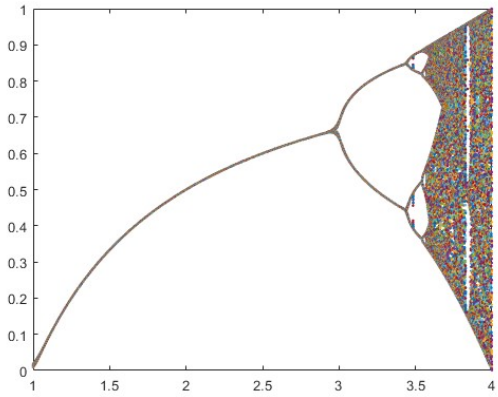

Figure 4.9: The bifurcation diagram is generated by iterating the logistic map for different values of the parameter α. The outer loop sweeps through r values from 1 to 3.999 in increments of 0.02. For each r, the logistic equation $y_{t+1} = ry_t(1 - y_t)$ is iterated for 100 time steps, starting from an initial condition $y = 0.3$. The results are plotted, with the x-axis representing the time step and the y-axis representing the value of y. This process visualizes how the dynamics change with different α, demonstrating behaviors such as fixed points, periodic cycles, and chaos. (For code see A.5)

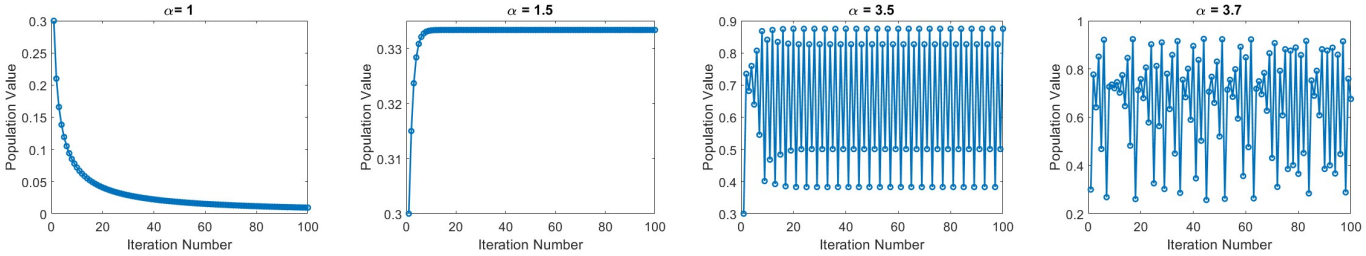

Figure 4.10: Plot of the logistic equation for specific values of the growth rate parameter α (1.0, 1.5, 3.5, and 3.7). Each curve represents the population dynamics over 100 iterations starting from an initial population size of 0.3. The logistic equation $y_{t+1} = ry_t(1 - y_t)$ is used to compute the population size at each iteration. The figure illustrates how different values of α lead to distinct behaviors in the population dynamics, with each curve labeled by its corresponding α value. (For code see A.5 with the specific values of α).

4.2.2 Topological Approach: Persistent Homology

To generate the relative bifurcation diagram using the TDA approach (A.6), we follow the same process as discussed in Section 4.1.2. And the resulting plot is in figure 4.11.

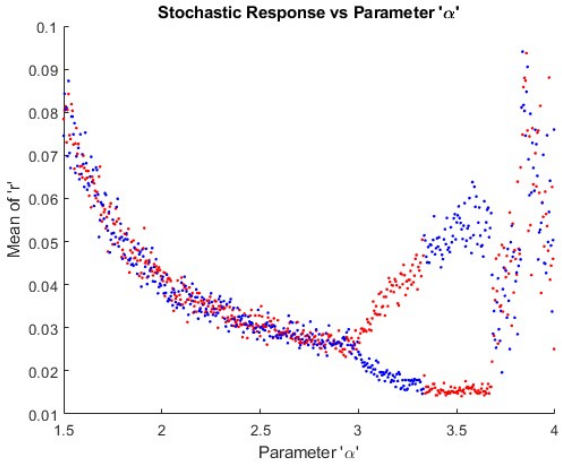

Figure 4.11: Bifurcation diagram illustrating the stochastic response of the logistic equation as a function of the growth rate parameter α. The diagram is generated using a combination of the logistic map and a stochastic model. The logistic map $y_{t+1} = ay_t(1 - y_t)$ is iterated 100 times starting from an initial population size of 0.3. For each value of α in the range [1.5, 3.999] with a step size of 0.005, the last 50 iterations are used to compute the stochastic response using the function `stochasticode`. The mean values of the stochastic response for odd- and even-indexed elements are plotted as red and blue dots, respectively. This diagram reveals the complex behavior and bifurcations in the system as the parameter α varies. (For code see at A.6)

4.2.3 Comparison and Results

Virtual distribution

Now we proceed with the comparison of the two bifurcation diagrams produced with he classical approach, using directly the discrete problem (4.2), and via the topological approach.

Observations

Several key observations can be drawn from these plots:

1. The lower plot, generated using TDA, accurately mirrors the form of the bifurcation diagram produced by the classical approach, up to the phase where chaotic behavior appears. This is a consequence of the fact that when the activation population levels, predicted by the discrete system (4.2), increase, the minimum radius at which the Betti-1 number emerges becomes smaller, as indicated in the filtration process in Figure A.3. This demonstrates the effectiveness of TDA tools in capturing the essential dynamical behavior of the system. The emergence and disappearance of topological features in the lower plot correspond to the bifurcation points and the transitions between different dynamical regimes observed in the classical approach.

2. In the lower plot, regions with tightly clustered points correspond to the periodic behavior observed in the classical bifurcation diagram. Conversely, regions with more scattered points in the lower plot indicate chaotic behavior, mirroring the chaotic regions in the upper plot. This underscores the ability of TDA to distinguish between different dynamical behaviors based on topological features.

3. The lower plot offers additional quantitative insights into the system's dynamics. The magnitude of the topological features, quantified by their persistence, diminishes as the system transitions

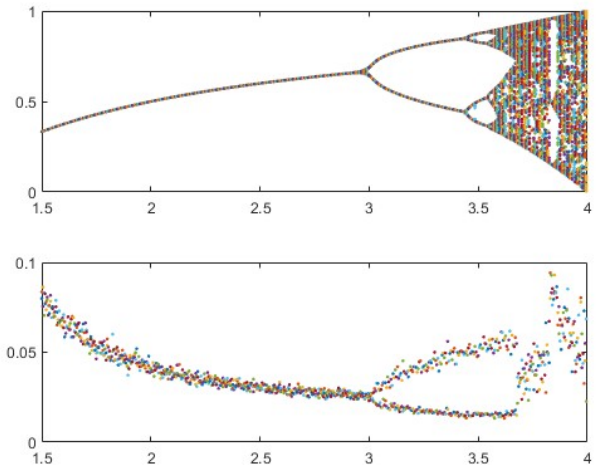

Figure 4.12: Bifurcation diagrams illustrating the behavior of the logistic equation under both classical and stochastic approaches. The upper diagram represents the classical approach, while the lower diagram depicts the stochastic response. In the stochastic approach, for each value of the parameter a in the range [1.5, 3.999] with a step size of 0.005, the last 50 iterations of the logistic map $y_{t+1} = ay_t(1 - y_t)$ are used to compute the stochastic response using the function `stochasticode`. The mean values of the stochastic response for odd- and even-indexed elements are plotted as red and blue dots, respectively. This comparison highlights the differences and similarities in the system's dynamics under deterministic and stochastic influences. The second plot with topology, influenced by stochasticity, does not clearly show the transition from period 2 to 4 due to stochastic fluctuations. We are still working on improving it. (For Code see A.6).

into chaos. This implies that chaotic regions are marked by less pronounced topological features, a characteristic that could be instrumental in quantifying the complexity of the system's behavior.

4.3 Coefficient of Variance Test

The coefficient of variance (CV) measures the *relative variability* (relative variability is essential for studying the dynamics of systems, as it offers valuable insights into their behavior, stability, and responsiveness to external factors.) of a dataset compared to its mean. It is calculated as the ratio of the *standard deviation* to the *mean* of the dataset.

In our first case, we calculate the coefficient of variance (CV) for $d(t)$ time series in two different bi-stable situations. Module1 is referring the upper stability (for any $d(t) > 0.5$) in Figure 4.2) while Module2 is for lower one (for any $d(t) < 0.5$) in Figure 4.2). Since there are no bi-stable situations for the $r_{\min}(t)$ dataset, we compute the CV for $r_{\min}(t)$ based on the time intervals corresponding to the bi-stability periods of $d(t)$. In the second case, we calculate the overall CV for both data sets.

- Since the CV values for $d(t)$ is relatively low, it indicates that the variability of the data points around the mean is relatively small. This suggests that the values in the $d(t)$ dataset are more tightly clustered around the mean, indicating a lower degree of dispersion or variability.

CV Type	d(t)	$r_{\min}(t)$
Module1	0.36949	0.69038
Module2	0.12033	0.40664
Overall	0.38655	0.64321

Model 1 (SDE)

CV Type	d(t)	$r(t)$
Overall	0.72127	1.41667

Model 2 (Discrete Logistic)

- The higher CV values for $r_{\min}(t)$ variability of the data points around the mean is relatively high. This indicates a larger degree of dispersion or variability in the values of the $r_{\min}(t)$ dataset compared to its mean.

Chapter 5

Conclusions and Future Works

In this closing chapter we draw the main conclusions of our study as well as we propose some future research directions.

5.1 Main Conclusions

1. We investigated the stability behavior of a stochastic differential equation (SDE) model characterized by a double-well potential drift term using two distinct methodologies. Initially, we employed a classical approach by discretizing the model through the Euler-Maruyama approximation scheme, thereby generating the first time series. Subsequently, we applied topological data analysis (TDA) to derive a second time series, which was defined by the minimum radius at which the first Betti number (Betti 1) emerges. Our observations revealed that both time series exhibited similar stability behavior, specifically a bi-stable behavior, for the underlying stochastic model, see Figure 4.8. Therefore, leveraging TDA tools provides an alterantive way to study the bi-stability of stochastic system (4.1).

2. Next, we analysed a discrete logistic model (4.2), which displays considerable variability in its dynamics. The system's behaviour, contingent on the bifurcation parameter's value, spans from converging to a unique equilibrium, to periodic doubling, and even exhibits chaotic behaviour. By investigating the temporal evolution of the Betti 1 number, we can track all types of dynamics, barring the chaotic behaviour. For a visual representation of these dynamics, refer to Figure 4.12.

5.2 Future Works

While this research has provided insights into the application of TDA, several avenues remain for future exploration:

1. **Noise reduction analysis:** Conduct a comprehensive study to explore the conditions under which the topological approach might effectively reduce noise in data sets. This could involve comparing different types of noise and evaluating the robustness of topological methods.

2. **Non-homogeneous bi-stability:** Investigate in greater detail the impact of non-homogeneous bi-stability on data sets. This includes studying the dynamic behavior of normalised activity datasets and identifying specific changes induced by stochastic systems exhibiting bi-stability.

3. **Chaotic behavior of discrete difference equations:** Explore the chaotic behavior of discrete difference equations using topological data analysis as in [8]. This investigation can reveal new insights into the underlying structure of chaotic systems and enhance the understanding of complex dynamical behaviors through topological methods.

By pursuing these directions, future research can build on the findings of this thesis and contribute to the advancement of topological methods in data analysis.

List of Figures

Appendix A

Supplementary Material

This appendix provides supplementary materials to complement the main content of the thesis. Here, readers will find additional information, including code snippets, software tools utilized in the analysis, and figures referenced throughout the document. The materials presented in this section aim to enhance understanding, facilitate reproducibility, and provide further insights into the methodologies employed in the research.

A.1 Codes for Figures 4.1 and 4.2

To produce the data, we first create two separate function files in MATLAB, each explaining aspects of our model: `saddle.m`(see at A.1) and `EulMarih2.m`(see at A.1). The `saddle.m` file introduces the non-linearity of our function, while the `EulMarih2.m` file incorporates the Euler-Maruyama method by combining the function from `saddle.m` with the time series. Then, we create a main file named `mainstoch.m` (see at A.1) to combine this two file where we rename "X(t)" as "y".
Then, we run the main file and the resulting plot can be seen in Figure 4.1.

EulMarih2.m

```
function
   [t1,y]=EulMarih2(f,tf,x)
n=length(x);
t=0;
flag=zeros(n,1);
h=0.01;
it=1;
sigma=2;
St=zeros(1,n);
while t<tf
     y(it,:)=x;
     t1(it)=t;
     f1=f(t,x);
     x=x+h*f1+sigma*randn(1,n)*sqrt(h);
     t=t+h;
     it=it+1;
end
end
```

saddle.m

```
function y=saddle(t,x)
y=4*x-x^3;
end
```

mainstoch.m

```
 % Load the necessary
    functions
load_javaplex;

% Generate the data sets
[t1, y] =
   EulMarih2(@saddle,
   100, 1);

% Normalize the first
   data set (dt)
dt = (y - min(y)) /
   (max(y - min(y)));

% Plot the first data
   set (y)
plot(t1, dt);
```

A.2 Code for Figure 4.5

```
    N = 100; % Number of equidistant points
theta = linspace(0, 2*pi, N); % Create an array of angles from 0 to 2*pi
   with N points
x = cos(theta); % Calculate x-coordinates of the unit circle
y = sin(theta); % Calculate y-coordinates of the unit circle

% Initialize a binary vector to indicate active points
active_points = rand(1, N) < 0.5; % randomly activate around half of the
   points

% Plot the unit circle
theta_circle = linspace(0, 2*pi, 100); % Create an array of angles for
   plotting the circle
x_circle = cos(theta_circle); % Calculate x-coordinates of the unit circle
y_circle = sin(theta_circle); % Calculate y-coordinates of the unit circle
plot(x_circle, y_circle, 'b'); % Plot the unit circle
hold on;

% Plot inactive points on the border of the unit circle with red color
plot(x(~active_points), y(~active_points), 'ro', 'MarkerSize', 5);

% Plot active points with bold color
plot(x(active_points), y(active_points), 'bo', 'MarkerSize', 5,
   'LineWidth', 2);

title('Unit Circle with Equidistant Points on Border');
xlabel('x');
ylabel('y');
axis equal;
grid on;
```

[language=Matlab]

A.3 Code for Figure 4.6

filtration.m

```
function filtration(x,r)
n=length(x);
for i=1:n
x0=x(i,1);
y0=x(i,2);

t = (1/16:1/32:1)'*2*pi;
x1=x0+r/2*cos(t);
y1=y0+r/2*sin(t);
fill(x1,y1,'r')
title(['\epsilon = ',num2str(r)])
set(gca,'fontsize',16)
end
hold on
end
```

homology.m

```
x=points_cloud;
figure(4)
for i=1:8
r=R(i);
subplot(1,4,i)
filtration2(x,r)
end
```

A.4 Codes for Figures 4.7 and 4.8

stochasticode2.m

```
    function r = stochasticode(t1, dt, n)
    import edu.stanford.math.plex4.*;

    t = 0:2*pi/n:2*pi-2*pi/n;
    x = cos(t);
    y = sin(t);

    r = zeros(length(dt), 1); % Initialize result vector

    for i = 1:length(dt)
        d = dt(i);
        index = randperm(n);
        indact = index(1:round(d*n)); % Select random indices

        if length(indact) > 1
            cour = [x(indact)', y(indact)'];
            num_landmark_points = min(length(indact), 50);

            % Limit to 50 if more than 50 points
            nu = 1;
            num_divisions = 1000;

            % Create the set of points
            point_cloud = cour;

            % Create a sequential maxmin landmark selector
            landmark_selector = api.Plex4.createMaxMinSelector(point_cloud,
               num_landmark_points);

            maxmin_points =
               point_cloud(landmark_selector.getLandmarkPoints() + 1, :);

            R = landmark_selector.getMaxDistanceFromPointsToLandmarks();
            max_filtration_value = 2 * R;
            max_dimension = 3;

            % Create a lazy witness stream
stream =
   streams.impl.LazyWitnessStream(landmark_selector.getUnderlyingMetricSpace(),
```

```
   landmark_selector, max_dimension,
37
38 max_filtration_value, nu, num_divisions);
39             stream.finalizeStream();
40
41             % Compute the intervals
42             persistence =
                  api.Plex4.getModularSimplicialAlgorithm(max_dimension, 2);
43             intervals = persistence.computeIntervals(stream);
44 A1 =
   edu.stanford.math.plex4.homology.barcodes.BarcodeUtility.getEndpoints(intervals,
   1, 0);
45
46             if ~isempty(A1)
47                 r(i) = A1(1);
48             else
49                 r(i) = 0;
50             end
51         else
52             r(i) = 0;
53         end
54     end
55 end
```

mainstoch.m

```
1     % Load the necessary functions
2 load_javaplex;
3
4 % Generate the data sets
5 [t1, y] = EulMarih2(@saddle, 100, 1);
6
7 % Normalize the first data set (dt)
8 dt = (y - min(y)) / (max(y - min(y)));
9
10 % Plot the first data set (dt)
11 plot(t1, dt);
12 hold on;
13
14 % Generate the second data set (r)
15 n = 10000;
16 r = stochasticode(t1, dt, n);
17
18 % Plot the second data set (r)
19 yyaxis right;
20 plot(t1, r);
21
22 % Add legend
23 legend('dt', 'r');
```

```

% Add labels for both data sets on the y-axis
ylabel('dt');
yyaxis right;
ylabel('r');
```

A.5 Code for Figure 4.9

logistic-euqtion.m

```
for r = 1:0.02:3.999
    y = 0.3;
    x = logistic_equat(r, y, 100);
    plot(x(:, 1), x(:, 2), 'o-',
       'LineWidth', 2);
    set(gca, 'fontsize', 16);
    title(num2str(r)); % Convert r
       to a string for the title
    pause;
end
```

main-log.m

```
function x = logistic_equat(r, y,
   Tfin)
    x = zeros(Tfin, 2);
    for i = 1:Tfin
        x(i, :) = [i, y];
        y = r * y * (1 - y);
    end
end
```

A.6 Code for Figure 4.11

To generate the relative bifurcation diagram using the TDA approach, we follow the same process as discussed in Section 4.1.2. We just call the MATLAB file `logistic-equation.m` in the stochastic code instead of `EulMarih2.m`.

stochastic.m

```
function r = stochasticode(dt, n)
import edu.stanford.math.plex4.*;

t = 0:2*pi/n:2*pi-2*pi/n;
x = cos(t);
y = sin(t);

r = zeros(1, length(dt)); %
   Pre-allocate r

for i = 1:length(dt)
    d = dt(i);
    index = randperm(n);
    indact = index(1:round(d*n));

    cour = [x(indact)', y(indact)'];
    if length(indact) > 50
        num_landmark_points = 50;
    else
        num_landmark_points =
           length(indact);
    end

    nu = 1;
    num_divisions = 100;

    if length(indact) > 1
        point_cloud = cour;
        landmark_selector =
           api.Plex4.createMaxMinSelector(po
           num_landmark_points);
        maxmin_points =
           point_cloud(landmark_selector.get
           + 1, :);

```

```
        R =
            landmark_selector.getMaxDistanceFromPointsToLandmarks();
        max_filtration_value = 6 * R;
        max_dimension = 3;
        stream =
            streams.impl.LazyWitnessStre
            landmark_selector,
            max_dimension,
            max_filtration_value, nu,
            num_divisions);
        stream.finalizeStream();

        persistence =
            api.Plex4.getModularSimplici
            2);
        intervals =
            persistence.computeIntervals
        A1 =
            edu.stanford.math.plex4.homo
            1, 0);
    else
        A1 = [];
    end

    if ~isempty(A1)
        r(i) = A1(1);
    else
        r(i) = 0;
    end
end
```

main-log.m

```
cells = cell(1, 3); % Pre-allocate
    cells array
k = 1;
for a = 1.5:0.005:3.999
    y = 0.3;
    x = logistic_equat(a, y, 100);
    cells{k, 1} = a;
    cells{k, 2} = x(50:end, 2);
    r = stochasticode(x(50:end, 2),
        1000);
    cells{k, 3} = r;
    k = k + 1;
end

figure;
hold on;
for i = 1:k-1
    a = cells{i, 1};
    r = cells{i, 3};

    plot(a, mean(r(1:2:end)), 'r.');
        % Plot mean of odd-indexed
        elements of r
    plot(a, mean(r(2:2:end)), 'b.');
        % Plot mean of even-indexed
        elements of r
end
xlabel('Parameter a');
ylabel('Mean of r');
legend({'Mean of odd-indexed r',
    'Mean of even-indexed r'});
title('Stochastic Response vs
    Parameter a');
hold off;
```

A.7 Code for Figure A.1

```
%% sampled from a figure 8.
clc; clear; close all;
import edu.stanford.math.plex4.*;

max_dimension = 3;
max_filtration_value = 1.1;
num_divisions = 1000;
% Select 75 random points from the figure 8 space.
point_cloud = examples.PointCloudExamples.getRandomFigure8Points(75);
```

```
scatter(point_cloud(:,1),point_cloud(:,2)), axis equal
% create a Vietoris-Rips stream
stream = api.Plex4.createVietorisRipsStream(point_cloud, max_dimension,
    max_filtration_value, num_divisions);
% get persistence algorithm over Z/2Z
persistence = api.Plex4.getModularSimplicialAlgorithm(max_dimension, 2);
% compute the intervals
intervals = persistence.computeIntervals(stream);
% create the barcode plots
options.filename = 'ripsFigure8';
options.max_filtration_value = max_filtration_value;
options.max_dimension = max_dimension - 1;
plot_barcodes(intervals, options);
```

A.8 Code For Figure A.2

```
clc; clear; close all;
import edu.stanford.math.plex4.*;
%% Figure 8 Example
% initialize the point cloud
point_cloud = examples.PointCloudExamples.getRandomFigure8Points(1000);
% create the landmark selectors
num_landmark_points = 100;
random_selector = api.Plex4.createRandomSelector(point_cloud,
    num_landmark_points);
maxmin_selector = api.Plex4.createMaxMinSelector(point_cloud,
    num_landmark_points);
% extract the subset of landmark points from the original point cloud
% Note: we need to increment the indices by 1 since Java uses 0-based
% arrays
random_points = point_cloud(random_selector.getLandmarkPoints() + 1, :);
maxmin_points = point_cloud(maxmin_selector.getLandmarkPoints() + 1, :);
% Plot the landmark points
subplot(1, 2, 1);
scatter(random_points(:,1), random_points(:, 2), '+');
title('Random landmark selection');
subplot(1, 2, 2);
scatter(maxmin_points(:,1), maxmin_points(:, 2), '+');
title('Maxmin landmark selection');
print('-depsc', 'landmarks.eps');
system('epstopdf landmarks.eps')
```

A.9 Javaplex in MATLAB

To install Javaplex for Matlab, one can visit the latest release at `https://github.com/appliedtopology/javaplex/releases/latest/`. Download the zip file containing the Matlab examples, which should be named something like `matlab-examples-4.2.3.zip`. Extract the contents of the zip file. The resulting

folder should be named `matlab-examples`; ensure you know the location of this folder and that it is not within the zip file.

In MATLAB, navigate to the directory containing the `matlab-examples` folder that you extracted from the zip file. In the MATLAB command window, run the `load_javaplex.m` file.

```
>> load_javaplex
```

Then, in the MATLAB command window, type the following command.

```
>> import edu.stanford.math.plex4.*;
```

Installation is now complete. Confirm that Javaplex is working properly with the following command.

```
>> api.Plex4.createExplicitSimplexStream()
```

Your output should resemble the following, with the last several characters being different:

```
ans = edu.stanford.math.plex4.streams.impl.ExplicitSimplexStream@513fd4
```

Each time you start a new MATLAB session, remember to run `load_javaplex.m`.

A.9.1 Computing Persistence Homology

The MATLAB script corresponding to this section is `explicit_simplex_example.m`, located in the `tutorial_examples` folder. You can either copy and paste commands from this script into the MATLAB window, or you can run the entire script at once using the following command:

```
>> explicit_simplex_example
```

To build a simplicial complex in Javaplex we simply build a stream in which all filtration values are zero. First we create an empty explicit simplex stream. Next, we add simplices using the methods `addVertex` and `addElement`. After we are done building the complex, calling the method finalizeStream is necessary before working with this complex! For Example:

```
>> stream.addVertex(0);
>> stream.addVertex(1);
>> stream.addVertex(2);
>> stream.addElement([0, 1]);
>> stream.addElement([0, 2]);
>> stream.addElement([1, 2]);
>> stream.finalizeStream();
```

Now, we can create an object that will compute the homology of our complex. The first input parameter 3, indicates that homology will be computed in dimensions 0, 1, and 2: that is, in all dimensions strictly less than 3. The second input, 2, means that we will compute homology with $\mathbb{Z}/2\mathbb{Z}$ coefficients, and this input can be any prime number.

```
>> persistence = api.Plex4.getModularSimplicialAlgorithm(3, 2);
```

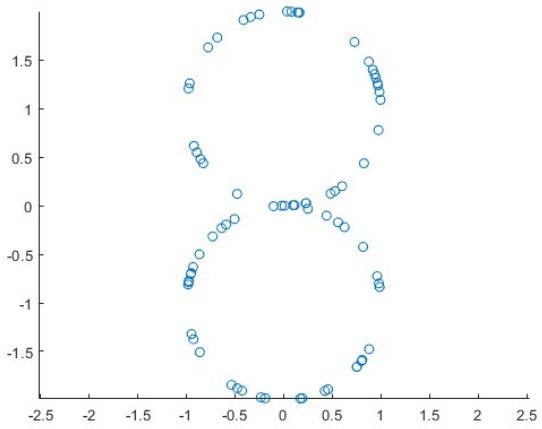

Figure A.1: Vietoris–Rips complex of 75 points randomly sampled from a "figure 8" space. See code at A.7

A.9.2 Vietoris–Rips Streams

The MATLAB script corresponding to this section is `rips_example.m`. For example: to compute the persistent homology of the Vietoris–Rips complex of 75 points randomly sampled from a "figure 8" space we can use this script A.7 in Javaplex. The resulting plot is in figure A.1:

A.9.3 Landmark Selection

There are two common methods for selecting landmark points. The first is to choose the landmarks L randomly from the point cloud Z. The second is a greedy inductive selection process called sequential maxmin. In sequential maxmin, the first landmark is picked randomly from Z. Inductively, if L_{i-1} is the set of the first $i-1$ landmarks, then let the i-th landmark be the point of Z which maximizes the function $z \mapsto d(z, L_{i-1})$, where $d(z, L_{i-1})$ is the distance between the point z and the set L_{i-1}.

The MATLAB script corresponding to this section is `landmark_example.m`. Here is the script (A.8) for showing the difference between randomized and maxmin landmark. The resulting plot is:

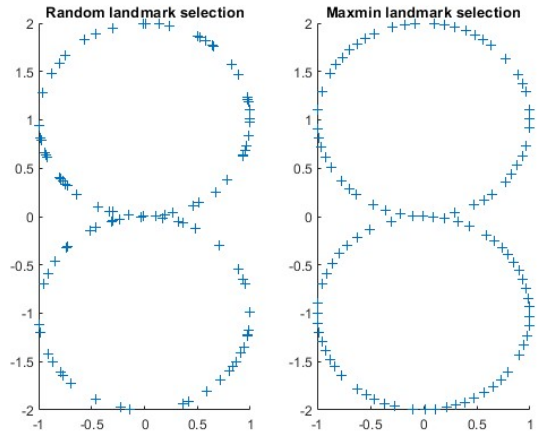

Figure A.2: Left: Random Landmark Selection, Right: Maxmin Landmark Selection. See code at A.8

Bibliography

[1] Jean-Daniel Boissonnat, Frdric Chazal, and Mariette Yvinec. *Geometric and Topological Inference*. Cambridge University Press, USA, 2018.

[2] Crispin Gardiner. *Stochastic Methods*, volume 4. Springer Berlin, 2009.

[3] Peter Hänggi, Peter Talkner, and Michal Borkovec. Reaction-rate theory: fifty years after kramers. *Reviews of Modern Physics*, 62(2):251, 1990.

[4] Desmond J. Higham and Peter E. Kloeden. *An Introduction to the Numerical Simulation of Stochastic Differential Equations*. SIAM, 2010.

[5] Robert M May. Simple mathematical models with very complicated dynamics. *Nature*, 261(5560):459–467, 1976.

[6] Kazuo Nishimura and Makoto Yano. Discrete-time in economics: An introduction. *Journal of Difference Equations and Applications*, 10(5):449–451, 2004.

[7] Andrew Tausz and Mikael Vejdemo-Johansson. *Javaplex: A Research Software Package for Persistent (Co)Homology*. Springer Verlag, 2014.

[8] Joshua R Tempelman and Firas A Khasawneh. A look into chaos detection through topological data analysis. *Physica D: Nonlinear Phenomena*, 406:132446, 2020.

[9] Nicolaas Godfried Van Kampen. *Stochastic Processes in Physics and Chemistry*, volume 1. Elsevier, 1992.

[10] Pierre-François Verhulst. *Deuxième mémoire sur la loi d'accroissement de la population*, volume 269. Hayez, 1847.