Chaos and Time Series Analysis: Optimization of the Poincaré Section and distinguishing between deterministic and stochastic time series

Jeremy George Cavers

University of Windsor

Follow this and additional works at: https://scholar.uwindsor.ca/etd

Recommended Citation

https://scholar.uwindsor.ca/etd/7242
CHAOS AND TIME SERIES ANALYSIS:

OPTIMIZATION OF THE POINCARÉ SECTION

AND DISTINGUISHING BETWEEN

DETERMINISTIC AND STOCHASTIC TIME SERIES

by

Jeremy George Cavers

A Thesis
Submitted to the Faculty of Graduate Studies
through Mathematics and Statistics
In Partial Fulfillment of the Requirements for
The Degree of Master of Science at the
University of Windsor

Windsor, Ontario, Canada

© 2017 Jeremy George Cavers
CHAOS AND TIME SERIES ANALYSIS:
OPTIMIZATION OF THE POINCARÉ SECTION
AND DISTINGUISHING BETWEEN
CHAOTIC AND STOCHASTIC TIME SERIES

by
Jeremy George Cavers

Approved by:

D. Wu
School of Computer Science

M. Hlynka
Department of Mathematics and Statistics

R. J. Caron, Co-advisor
Department of Mathematics and Statistics

R. Gras, Co-advisor
School of Computer Science

July 12, 2017
Author’s Declaration of Originality

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication.

I certify that, to the best of my knowledge, my thesis does not infringe upon anyone’s copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances in my appendix.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.
Abstract

This thesis is concerned with chaos theory and the analysis of time series using the Poincaré and Higuchi (P&H) method. The P&H method has been shown to qualitatively differentiate between deterministic and stochastic time series. This thesis proposes that the P&H method can be extended to also quantitatively differentiate between deterministic and stochastic time series. This extension of the P&H method was tested on twelve time series: six produced by deterministic chaotic systems and six produced by stochastic processes. Results show that, even with noise, the P&H method can quantitatively differentiate between these two sets of time series.

This thesis also studies the problem of optimizing the location of the Poincaré section used in the P&H method. Proposed optimization methods were tested on the same twelve time series. Of the methods tested, the most effective Poincaré sections were found by a local search method.
Dedication

To my parents, for providing me with the love, means and opportunities to flourish as an individual.
Acknowledgments

I first extend my most sincere thanks to my co-supervisors Dr. Richard J. Caron and Dr. Robin Gras; their guidance made my research a most interesting and enjoyable experience. Their commitment to myself and this research project can not be overstated.

I would like to thank Paul Preney and Robert Mavrinac for their much needed support in computer programming. Paul is a remarkable resource in programming using C++. He also went very far out of his way to ensure our research team’s program became a successful tool.

I am most grateful to Drs. Caron, Gras, Monfared, Hu, Yang, and Justin Lariviere for their adept teaching of higher mathematics and computer science. They fostered an amazing learning environment where I could truly excel. In particular, I would like to thank Justin Lariviere for being most helpful and willing to answer any questions in all areas of mathematics.

I would like to thank the members of my defense committee. In particular, I am quite grateful to Dr. Dan Wu and Dr. Hlynka for agreeing to evaluate my work as external program readers, and to Dr. Tim Traynor for chairing the defense.

Dr. Krzysztof Michalak of Wroclaw University of Economics has been most generous to make the programs he developed for the P&H Method available online for future research. He has also offered many other resources of data and explanations in relation to his work on this method.

Finally, I thank my family, without their support this research simply could not have been completed. My wife, Bubby, for your unparalleled support; thank you.
# Contents

Author’s Declaration of Originality ........................................................... iii
Abstract ................................................................................................................ iv
Dedication ............................................................................................................. v
Acknowledgments ............................................................................................... vi
List of Figures ......................................................................................................... xi
Listings ................................................................................................................ xvi

1 Introduction ........................................................................................................ 1
   1.1 Overview and Outline .................................................................................... 1
   1.2 Dynamical Systems vs. Stochastic Processes ................................................. 3
   1.3 Notation and Definitions ................................................................................ 5
       1.3.1 One-Dimensional Maps ........................................................................ 5
       1.3.2 Phase Space Methods ........................................................................... 9
       1.3.3 Chaotic Orbits, Attractors and Systems ............................................... 11
       1.3.4 Time-delay Embeddings ...................................................................... 14
       1.3.5 Takens Delay Embedding Theorem ..................................................... 15

2 The Poincaré and Higuchi (P&H) Method ......................................................... 16
   2.1 The Poincaré Section .................................................................................... 16
       2.1.1 Intersection Method (Σ_{int}) ................................................................ 18
       2.1.2 Slice Method (Σ_{slice}) .................................................................... 18
       2.1.3 Poincaré Map .................................................................................... 19
   2.2 The Higuchi Fractal Dimension ................................................................... 20
   2.3 The P&H Method ....................................................................................... 23
3 Time Series Data Sets

3.1 Stochastic Time Series

3.1.1 Gaussian (Normal) 1D

3.1.2 Gaussian 3D

3.1.3 Random Walk 1D

3.1.4 Random Walk 3D

3.1.5 Uniform 1D

3.1.6 Uniform 3D

3.2 Deterministic Time Series

3.2.1 Logistic map

3.2.2 Bouali

3.2.3 Chen-Lee

3.2.4 Lorenz

3.2.5 Rabinovich-Fabrikant

3.2.6 Rössler

3.3 Adding Gaussian White Noise to Data Sets

4 Optimization of the Poincaré Plane in $\mathbb{R}^3$

4.1 Current Objective Function

4.2 New Objective Function

4.3 Summary of P&H Method and Objective Function

5 Previous Optimization Methods

5.1 Coordinate Axes

5.2 Principle Component Analysis (PCAm in/PCA max)

5.3 Gradient

5.4 Evolutionary Algorithm

6 Analysis of the Solution Space

6.1 Sphere Gradient Map

6.2 Flow Gradient Map

6.3 Normal Gradient Map
CONTENTS

Vita Auctoris \hspace{1cm} 114
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1 (a)</td>
<td>Intersection method to obtain $\hat{\Sigma}_{int} = {s_1, s_2, s_3, \ldots, s_N}$.</td>
</tr>
<tr>
<td>2.1 (b)</td>
<td>The Poincaré Map mapping successive intersection points.</td>
</tr>
<tr>
<td>2.2 (a)</td>
<td>Slice method to obtain $\hat{\Sigma}_{slice} = {s_1, s_2, s_3, \ldots, s_N}$.</td>
</tr>
<tr>
<td>2.2 (b)</td>
<td>The Poincaré Map mapping successive intersection points.</td>
</tr>
<tr>
<td>2.3</td>
<td>Obtained pattern from the P&amp;H method’s output for all chaotic time series. Source: [9]</td>
</tr>
<tr>
<td>2.4</td>
<td>The smooth decreasing trend of the P&amp;H method’s outputs for stochastic time series. Source: [9]</td>
</tr>
<tr>
<td>3.1</td>
<td>Collection of attractors for embeddings of the of the Logistic map into $\mathbb{R}^3$, with lag $\tau = 1$ and parameter $r \in [3.6, 4.0]$.</td>
</tr>
<tr>
<td>3.2</td>
<td>Logistic map bifurcation diagram for the parameter $r \in [2.4, 4.0]$.</td>
</tr>
<tr>
<td>3.3</td>
<td>Logistic map embedded into $\mathbb{R}^3$ with lag $\tau = 1$ and parameters $r = 3.8284$ and $x_0 = 0.712765$.</td>
</tr>
<tr>
<td>3.4 (a)</td>
<td>Bouali system: chaotic attractor.</td>
</tr>
<tr>
<td>3.4 (b)</td>
<td>Bouali system: 1500 point time series.</td>
</tr>
<tr>
<td>3.5 (a)</td>
<td>Chen-Lee system: chaotic attractor.</td>
</tr>
<tr>
<td>3.5 (b)</td>
<td>Chen-Lee system: 1500 point time series.</td>
</tr>
<tr>
<td>3.6 (a)</td>
<td>Lorenz system: chaotic attractor.</td>
</tr>
<tr>
<td>3.6 (b)</td>
<td>Lorenz system: 1500 point time series.</td>
</tr>
<tr>
<td>3.7 (a)</td>
<td>Rabinovich-Fabrikant system: chaotic attractor.</td>
</tr>
<tr>
<td>3.7 (b)</td>
<td>Rabinovich-Fabrikant system: 1500 point time series.</td>
</tr>
<tr>
<td>3.8 (a)</td>
<td>Rössler system: chaotic attractor.</td>
</tr>
<tr>
<td>3.8 (b)</td>
<td>Rössler system: 1500 point time series.</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>4.1</td>
<td>Collection of functions used in the P&amp;H Method.</td>
</tr>
<tr>
<td>5.1</td>
<td>A basis obtained using the Principal Component Analysis (PCA) method for a two-dimensional set of points.</td>
</tr>
<tr>
<td>5.2</td>
<td>An example of the working of Michalak’s gradient descent optimization method.</td>
</tr>
<tr>
<td>5.3</td>
<td>A genotype used in Krzysztof Michalak’s evolutionary algorithm with the slice method.</td>
</tr>
<tr>
<td>5.4</td>
<td>An overview of the evolutionary algorithm Krzysztof Michalak implemented.</td>
</tr>
<tr>
<td>6.1</td>
<td>Sphere Gradient Map for the 1-D Gaussian Stochastic time series.</td>
</tr>
<tr>
<td>6.2</td>
<td>Zoom Figure 6.1: Sphere Gradient Map for the 3-D Uniform Stochastic time series.</td>
</tr>
<tr>
<td>6.3</td>
<td>Sphere Gradient Map for the 1-D Gaussian Stochastic time series.</td>
</tr>
<tr>
<td>6.4</td>
<td>Zoom Figure 6.3: Sphere Gradient Map for the 3-D Uniform Stochastic time series.</td>
</tr>
<tr>
<td>6.5</td>
<td>Sphere Gradient Map for the embedded Henon Chaotic Map.</td>
</tr>
<tr>
<td>6.6</td>
<td>Zoom Figure 6.5: Sphere Gradient Map for the embedded Henon Chaotic Map.</td>
</tr>
<tr>
<td>6.7</td>
<td>Sphere Gradient Map for the embedded Logistic Chaotic Map.</td>
</tr>
<tr>
<td>6.8</td>
<td>Zoom Figure 6.7: Sphere Gradient Map for the embedded Logistic Chaotic Map.</td>
</tr>
<tr>
<td>6.9</td>
<td>Sphere Gradient Map for Logistic Map, with no noise. Scale on right is the value of the objective function $\Delta$ (Eq. 4.1)</td>
</tr>
<tr>
<td>6.10</td>
<td>Sphere Gradient Map for Logistic Map, with noise $\eta = 0.1$. Scale on right is the value of the objective function $\Delta$ (Eq. 4.1)</td>
</tr>
<tr>
<td>6.11</td>
<td>Normal Gradient Map Gradient for Logistic Map, with noise $\eta = 0.1$. Scale on right is the value of the objective function $\Delta$ (Eq. 4.1)</td>
</tr>
<tr>
<td>6.12</td>
<td>Zoom: Normal Gradient Map Gradient for Logistic Map, with noise $\eta = 0.1$. Scale on right is the value of the objective function $\Delta$ (Eq. 4.1)</td>
</tr>
<tr>
<td>7.1</td>
<td>Flow Map of the Random Sample Method.</td>
</tr>
</tbody>
</table>
7.2 Flow Map of the Local Sphere Cap Search Algorithm. ................. 67
7.3 Cap of unit sphere in $\mathbb{R}^3$, denoted by $C_i$. ................. 68
7.4 Frame 1: Local Sphere Cap Search Algorithm converging for Logist Map. ................................................................. 71
7.5 Frame 2: Local Sphere Cap Search Algorithm converging for Logist Map. ................................................................. 72
7.6 Frame 3: Local Sphere Cap Search Algorithm converging for Logist Map. ................................................................. 73
8.1 Output of P&H Method using the “slice” method for the Chen-Lee time series; noise $\eta = 0.1$. .................................................. 75
8.2 Output of P&H Method for the Bouali time series; noise $\eta = 0.1$. . 76
8.3 Output of P&H Method for the Logistic Map; noise $\eta = 0.1$. . .. 76
8.4 Output of P&H Method for the Lorenz times series; noise $\eta = 0.1$. 77
8.5 Output of P&H Method for the Rabinovich-Fabrikant times series; noise $\eta = 0.1$. ................................................................. 77
8.6 Output of P&H Method for the Rössler times series; noise $\eta = 0.1$. 78
8.7 Output of P&H Method for the Guassian 1D times series. ............. 79
8.8 Output of P&H Method for the Guassian 3D times series. ............. 79
8.9 Output of P&H Method for the Uniform 1D times series. ............. 80
8.10 Output of P&H Method for the Uniform 3D times series. ............ 80
8.11 Output of P&H Method for the Random Walk 1D times series. ...... 81
8.12 Output of P&H Method for the Random Walk 3D times series. ...... 81
9.1 30 tests of the P&H Method on all 6 stochastic data sets using the local search (intersection) method and objective function $\Delta(n,p)$. . . 83
9.2 30 tests of the P&H Method on all 6 chaotic data sets (noise $\eta = 0.1$) using the local search method and objective function $\Delta(n,p)$. . . 84
9.3 30 tests of the P&H Method on all 6 chaotic data sets (no noise) using the local search method and objective function $\Delta(n,p)$. . . . . . 84
9.4 30 tests of the P&H Method on all 6 stochastic data sets using the local search method and objective function $\psi(n,p)$. ............... 85
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.5</td>
<td>30 tests of the P&amp;H Method on all 6 chaotic data sets (noise $\eta = 0.1$) using the <em>local search</em> method and objective function $\psi(n, p)$.</td>
</tr>
<tr>
<td>9.6</td>
<td>30 tests of the P&amp;H Method on all 6 chaotic data sets (no noise) using the <em>local search</em> method and objective function $\psi(n, p)$.</td>
</tr>
<tr>
<td>10.1</td>
<td>Summary of Results from Chapter 9.</td>
</tr>
<tr>
<td>D.1</td>
<td>Uniform distribution on $[0, 2\pi) \times [0, \phi_c)$. Source: [29].</td>
</tr>
<tr>
<td>D.2</td>
<td>Uniform distribution on $[0, 2\pi) \times [0, \phi_c)$ does not generate uniform points on $C$.</td>
</tr>
<tr>
<td>D.3</td>
<td>$dA$ of differential surface element in spherical coordinates.</td>
</tr>
<tr>
<td>D.4</td>
<td>Triangle in unit sphere underneath cap $C$.</td>
</tr>
<tr>
<td>D.5</td>
<td>Inverse Transform Sampling on $[0, 2\pi) \times [0, \phi_c)$.</td>
</tr>
<tr>
<td>D.6</td>
<td>Inverse Transform Sampling leads to uniformly distributed points on sphere cap $C$.</td>
</tr>
</tbody>
</table>
Listings

B.1 C++ Code generating pseudorandom points with Mersenne Twister . 97
B.2 Input .txt file for optimalcheck . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 98
Chapter 1

Introduction

1.1 Overview and Outline

Chaotic time series analysis and forecasting is of great importance for many applications including, but not limited to: weather determination, monitoring EEG signals of epileptic patients and financial market analysis.

The Poincaré and Higuchi (P&H) method was introduced by Golestani et al. [9]. In brief, if we consider a chaotic system in $\mathbb{R}^3$, the P&H method intersects the system with a chosen hyperplane, termed the Poincaré section. The method then applies the Higuchi dimension, with modifications, to produce a graph. From the pattern produced by the graph, we can qualitatively distinguish between deterministic and stochastic time series. This thesis proposes that the P&H method can also quantitatively distinguish between deterministic and stochastic time series.

Golestani and Gras have shown that the P&H Method can successfully distinguish between deterministic and stochastic time series when applied to biomedical data [10] and population simulations [8]. Golestani et al. [9], however, did not provide a method of selecting a quantitative, distinguishing value from the P&H
Method, nor a method of selecting the Poincaré section.

Krzysztof Michalak showed that in using the P&H Method, the plane must be well chosen in order to distinguish between stochastic and deterministic time series \[18\]. He applied five optimization methods for finding such a plane in \( \mathbb{R}^3 \). Of his five methods, two were able to correctly distinguish between some well defined chaotic and stochastic time series (data sets given in Section 3.2). In this thesis, we build upon these methods of Michalak and Golestani et al. and apply two new optimization methods for choosing the Poincaré Section that allow us to quantitatively distinguish between deterministic and deterministic time series.

The principle motivation for this work is to make long-term time series forecasting a reality for chaotic systems. Golestani and Gras \[7\] presented such a long-term time series forecasting method, GenericPred, based upon the P&H method. It requires a measure of the level of chaos of a time series as a reference to predict future values of the series. This measurement is what prompted the added quantitative criterion to the P&H Method.

We end the introduction with a review of some definitions and concepts related to chaos theory that lead to a better understanding to the series and methods discussed. Chapter 2 gives the details of the P&H Method. Chapter 3 provides the deterministic and stochastic time series data sets that were used in our experiments. Chapter 4 states the optimization problem to be solved. In Chapter 5, we present current optimization methods for determining the Poincairé Section, as implemented by Krzysztof Michalak \[18\]. Chapter 6 shows some analysis of the solution space in order to give better understanding to the optimization problem and develop new
methods. In Chapter 7, we propose two new optimization methods: Random Sample and Local Sphere Cap Search. Results of experiments for comparing new P&H optimization methods against establish methods are given in Chapter 8. Results of experiments for the new local search P&H optimization method are given in Chapter 9. The new quantitative criterion of the P&H Method is given in Chapter 10. Conclusions, further research questions, and extensions to the results of this thesis are discussed in chapter 11.

The Appendices include the Runge Kutta Numerical Method, the C++ program, OptimalCheck, that was developed and used for all necessary analysis and phase space methods, and finally, methods on how to generate uniform points on the unit sphere and on spherical caps.

1.2 Dynamical Systems vs. Stochastic Processes

We first define dynamical systems and stochastic processes because chaotic time series are indeed in a class of dynamical systems, and stochastic time series are in a class of stochastic processes.

**Definition 1.2.1** (Dynamical System). A dynamical system is a set of deterministic functions, equations or rules that specify the present state in terms of past states [2].

Note the use of the term deterministic. Chaotic systems are a subset of dynamical systems and hence by definition are deterministic. The term chaotic may be misleading, as in lay terms it refers to a system having no ordered structure. In chaos theory, however, it refers to a system having a very ordered structure. Although a chaotic system may have complex behaviour and be difficult to predict, it is still
1.2. DYNAMICAL SYSTEMS VS. STOCHASTIC PROCESSES

deterministic and is completely ordered according to the functions, equations, rules, and previous states, as mentioned above.

All chaotic time series we consider (Section 3.2) are 1-dimensional iterative maps or 3-dimensional dynamic flows (Def. 1.3.26) defined by differential equations. Their initial conditions (vectors/points) determine all later states and any state lies on their respective chaotic attractor.

**Definition 1.2.2** (Stochastic Process). A stochastic (or random) process is a collection of random variables indexed by time. A discrete time stochastic process \( X = \{ X_t \mid t \in \mathbb{N}_0 \} \) is a countable collection of random variables indexed by the non-negative integers. A continuous time stochastic process \( X = \{ X_t \mid 0 \leq t < \infty \} \) is an uncountable collection of random variables. In general, for any indexing set \( \mathcal{T} \subseteq \mathbb{R} \) having infinite cardinality, \( \{ X_t \}_{t \in \mathcal{T}} \) is a stochastic process where \( X_t \) is a random variable, \( \forall t \in \mathcal{T} \). If the cardinality of \( \mathcal{T} \) is finite, then \( X \) is not considered a stochastic process, but rather a random vector. [2]

According to the definition above, since all stochastic time series data sets we consider (Section 3.1) have cardinality \( |\mathcal{T}| = 1500 \), they are random vectors that are a subset of a discrete time stochastic process.
1.3 Notation and Definitions

We adopt some notation and definitions used by Krzysztof Michalak [18], Julien Clinton Sprott [30], and Alligood et al. [2]. This basic introduction of chaos theory is needed to understand the definition of a chaotic time series and chaotic dynamical flows.

1.3.1 One-Dimensional Maps

These definitions give understanding to how chaotic behaviour can arise even out of simple one-dimensional maps. We are specifically interested in the 1-dimensional Logistic map.

An important concept is that when one observes chaotic motion, it can be described as “stable in the large” (it attracts a large set of initial conditions) while “locally unstable” (it has a chaotic orbit). [2]

Definition 1.3.1 (Smooth function). A smooth function is one for which derivatives of all orders exist and are continuous functions.

Definition 1.3.2 (Map). A function whose domain and codomain are the same space is called a map.

Definition 1.3.3 (Unimodal Maps). A map $f(x)$ is unimodal if it has only one critical point, i.e. a point for which $f'(x) = 0$ or where the derivative does not exist.

Definition 1.3.4 (Conjugate Maps). The maps $f$ and $g$ are conjugate if they are related by a continuous one-to-one change of coordinates. That is, there exists a continuous one-to-one map $C$ such that: $C \circ f = g \circ C$. 
1.3. NOTATION AND DEFINITIONS

The existence of *Conjugate maps* is very important because a map $f$, whose Lyapunov exponents (see Definition 1.3.14) and other properties are difficult to calculate, may be conjugate to a simpler map $g$. Most properties of the map $g$, including its Lyapunov exponent, are invariant under the transformation of a conjugate map.

**Example 1.3.1.** The Lyapunov exponent (see Definition 1.3.14) for the Logistic map (see Section 3.2) is difficult to calculate because the map does not have constant derivatives. The Tent Map, defined as $T_\mu(x) : (0, 1) \rightarrow (0, 1)$ such that:

$$T_\mu(x) = \begin{cases} 
\mu x & 0 \leq x \leq \frac{1}{2} \\
\mu(1 - x) & \frac{1}{2} \leq x \leq 1
\end{cases}$$

The Logistic map is indeed conjugate to the Tent map under the conjugate map:

$$C(x) = \frac{1 - \cos(\pi x)}{2}$$

Since the Tent map has a constant derivative for all values, and its Lyapunov exponent can be easily calculated. Under the conjugate map $C$ above, the Logistic map has the same Lyapunov exponent.

**Definition 1.3.5** (Fixed point). A point $p$ is a fixed point of the map $f$ if $f(p) = p$.

**Definition 1.3.6** (Orbit). Let $x$ be a point and let $f$ be a map. The orbit of $x$ under $f$ is the set of points \{x, f(x), f^2(x), \ldots\}. The starting point $x$ for the orbit is called the initial value of the orbit.

**Definition 1.3.7** (Periodic points/orbits). Let $f$ be a map on $\mathbb{R}$. We call $p$ a periodic point of period $k$ if $k$ is the smallest positive integer such that $f^k(p) = p$. The orbit \{p, f(p), f^2(p), \ldots, f^{k-1}(p)\} is called a periodic orbit of the point $p$ and has period $k$. 
In this case, we use the abbreviated terms period-k point to describe \( p \) and period-k orbit to describe the periodic orbit of \( p \).

**Example 1.3.2.** Note that a period-k point \( p \) is a fixed point for \( f^k(x) \) since \( f^k(p) = p \).

**Definition 1.3.8** (Asymptotically periodic). Let \( f \) be a smooth map. An orbit \( \{x_1, x_2, \ldots, x_n, \ldots\} \) is called asymptotically periodic if it converges to a periodic orbit as \( n \to \infty \); this means that there exists a periodic orbit \( \{y_1, y_2, \ldots, y_k, y_1, y_2, \ldots\} \) such that:

\[
\lim_{n \to \infty} |x_n - y_n| = 0
\]

**Definition 1.3.9** (Neighbourhood). Let \( f \) be a map and \( x_0 \in \mathbb{R} \). A neighbourhood of \( p \) with radius \( \varepsilon > 0 \) is given by:

\[
N_\varepsilon(x_0) = \{x \in \mathbb{R} : |x - x_0| < \varepsilon\}
\]

**Definition 1.3.10** (Sink/Attracting fixed point). Let \( f \) be a map and \( p \) be a fixed point of \( f \). If all points sufficiently close to \( p \) are attracted to \( p \), then \( p \) is called a sink or an attracting fixed point. More precisely, \( p \) is a sink if:

\[
\exists \varepsilon > 0, \forall x \in N_\varepsilon(p), \lim_{k \to \infty} f^k(x) = p
\]

**Definition 1.3.11** (Basin). Let \( f \) be a map and \( p \) be a sink of \( f \). The set of all initial conditions whose orbits converge to \( p \) is the basin of the sink \( p \). More precisely:

\[
sink(p) = \{x : \lim_{k \to \infty} f^k(x) = p\}
\]

**Definition 1.3.12** (Source/Repelling fixed point). Let \( f \) be a map and \( p \) be a fixed point of \( f \). If all points sufficiently close to \( p \) are repelled from \( p \), then \( p \) is called a
1.3. NOTATION AND DEFINITIONS

source or an repelling fixed point. More precisely, \( p \) is a source if:

\[
\exists \varepsilon > 0, \forall x \in N_\varepsilon(p) \setminus \{p\}, \lim_{k \to \infty} f^k(x) \neq p
\]

**Definition 1.3.13** (Sensitive point). Let \( f \) be a map on \( \mathbb{R} \). A point \( x_0 \) has sensitive dependence on initial conditions if there is a nonzero distance \( \delta \) such that some points arbitrarily near \( x_0 \) are eventually mapped at least \( \delta \) units from the corresponding image of \( x_0 \). More precisely:

\[
\exists \delta > 0, \exists x \in \mathbb{R}, \exists k \in \mathbb{N} \ni \exists: |f^{k}(x) - f^{k}(x_0)| \geq \delta
\]

We will call such a point \( x_0 \) a sensitive point.

**Definition 1.3.14** (Lyapunov number and Lyapunov exponent). Let \( f \) be a smooth map of the real line \( \mathbb{R} \). The Lyapunov number \( L(x_1) \) of the orbit \( \{x_1, x_2, x_3, \ldots \} \) is:

\[
L(x_1) = \lim_{n \to \infty} \left( \prod_{i=1}^{n} |f'(x_i)| \right)^{\frac{1}{n}}
\]

if this limit exists. The Lyapunov exponent \( h(x_1) \) is defined as:

\[
h(x_1) = \ln(L(x_1)) = \lim_{n \to \infty} \frac{1}{n} \sum_{i=1}^{n} \ln |f'(x_i)|
\]

if this limit exists. Notice that \( h \) exists if and only if \( L \) exists and is nonzero. \([2]\)
1.3.2 Phase Space Methods

The following definitions are needed for understanding phase space methods we use, including the concept of embedding of a time series from the state space into the time-delay embedding space.

**Definition 1.3.15 (State space).** The state space (or configuration space) is the space of the dynamical variables. The dimension of the configuration space, denoted by $n_s$, is equivalent to the number of initial conditions that must be specified.

**Example 1.3.3.** The Hénon Map has a 2-dimensional state space ($n_s = 2$) with dynamic variables $x$ and $y$. It requires two initial conditions: $x_0$ and $y_0$.

**Example 1.3.4.** A Logistic Map has a 1-dimensional state space ($n_s = 1$) with the dynamic variable $x$, and initial condition $x_0$.

Note that in this paper we only consider time series with $n_s \in \{1, 3\}$.

**Definition 1.3.16 (Discrete time interval).** For a discrete time series $S_N = \{x_1, \cdots, x_N\}$, points in the series are sampled using time-intervals $\Delta t$. Given an initial condition $x_0$, each discrete point is given by $x_i = x(i\Delta t)$ for $1 \leq i \leq N$.

**Definition 1.3.17 (Time-delay embedding lag).** The time-delay embedding lag, or simply lag ($\tau$), is the number of time steps used to embed a series into the embedding space.

Note that in all our embeddings we use a lag of $\tau = 1$.

**Definition 1.3.18 (Embedding).** For a given one-dimensional time series $S_N = \{x_1, \cdots, x_N\}$, the embedding of $S_N$ into a space of greater dimension $n$, using a lag
of \( \tau \), is given by the set:

\[
\Gamma = \{(x_i, x_{i+\tau}, x_{i+2\tau}, \ldots, x_{i+(n-1)\tau}) : i = 1, \ldots, N - (n - 1)\tau \}
\]

**Definition 1.3.19** (Time-delay reconstructed state space). The time-delay reconstructed state space is a space obtained by embedding a time series into a higher dimension. We denote the dimension of the time-delay reconstructed state space by \( n_r \). The dimension of this space can be chosen arbitrarily to be \( n_r \in \{2, \cdots, N\} \) for a time series \( S_N = \{x_1, \cdots, x_N\} \).

**Definition 1.3.20** (Time-delay embedding space). The time-delay embedding space is a reconstructed state space chosen with the minimum dimension for which important dynamical and topological properties are preserved. We denote the dimension of the time-delay embedding space by \( n_e \).

In Section 1.3.5, we see that Takens Delay Embedding Theorem ensures the time-delay embedding space has dimension \( n_e = 2n_s + 1 \).
1.3. NOTATION AND DEFINITIONS

1.3.3 Chaotic Orbits, Attractors and Systems

**Definition 1.3.21** (Chaotic orbit). Let $f$ be a map of the real line $\mathbb{R}$. Let $\{x_1, x_2, x_3, \ldots\}$ be a bounded orbit of $f$. The orbit is chaotic if:

(i) $\{x_1, x_2, x_3, \ldots\}$ is not asymptotically periodic.

(ii) the Lyapunov exponent $h(x_1)$ is greater than zero.

A chaotic attractor (Def. 1.3.23) is a “forward limit set”, which intuitively is the set of points that remain in the chaotic orbit after removing an arbitrarily large number of the initial points from the orbit.

**Definition 1.3.22** (Forward limit set). Let $f$ be a map and let $x_0$ be an initial condition. The forward limit set of the orbit $\{f^n(x_0)\}$ is the set:

$$\omega(x_0) := \{ x | \forall N \in \mathbb{N}, \forall \varepsilon > 0, \exists n > N \ni |f^n(x_0) - x| < \varepsilon \}$$

This means that a point is in the forward limit set if you can return arbitrarily closer to that point, infinitely many times, by moving further in the orbit. We say that an orbit $\{f^n(x_1)\}$ is **attracted** to the forward limit set $\omega(x_0)$ if $\omega(x_1) \subseteq \omega(x_0)$.

It is important to note from this definition that points in an orbit may or may not be contained in its forward limit set. The forward limit set may have no points in common with the orbit, as is the case with the forward limit set of an orbit converging to a sink. **2**

**Definition 1.3.23** (Chaotic attractor). A chaotic attractor is a forward limit set $\omega(x_0)$ of an orbit $\{f^n(x_0)\}$ satisfying the following three conditions:

(i) the orbit $\{f^n(x_0)\}$ is chaotic
1.3. NOTATION AND DEFINITIONS

(ii) $x_0 \in \omega(x_0)$

(iii) the basin of attraction of $\omega(x_0)$ has a non-zero mesure.

The first two conditions defines a chaotic set. In general, a chaotic set may not be a chaotic attractor. \[15\]

Although there is no agreed upon precise definition of a chaotic system, the following are properties of a chaotic system that most mathematicians agree on. \[11\]

(i) the solution/orbits must be aperiodic.

(ii) the solution/orbits must be bounded.

(iii) the solution/orbits must be exponentially sensitive to initial conditions.

(iv) the system must be non-linear (in the case of 1D maps, it must be non-invertible and in the case of O.D.E.’s the system must be atleast three dimensional).

**Definition 1.3.24** (Homeomorphism). A function $f : X \rightarrow Y$ between two topological spaces $(X,T_X)$ and $(Y,T_Y)$ is called a homeomorphism if it has the following properties:

(i) $f$ is a bijection (one-to-one and onto)

(ii) $f$ is continuous

(iii) the inverse function $f^{-1}$ is continuous ($f$ is an open mapping)

**Definition 1.3.25** (Manifold). A manifold is a topological space that locally resembles Euclidean space near each point. More precisely, each point of an $n$-dimensional manifold has a neighbourhood that is homeomorphic to the Euclidean space $\mathbb{R}^n$.

The dynamics of all 3-dimensional chaotic systems can be determined by manifolds in $\mathbb{R}^3$. All 3-dimensional dynamic flows (defined below) in Chapter 3 lie on their
1.3. NOTATION AND DEFINITIONS

respective chaotic attractor \cite{23}.

**Definition 1.3.26** (Dynamic flow). A 3-dimensional dynamic flow ($\Gamma$) is defined by three differential equations in three variables, producing continuous trajectories on its respective chaotic attractor.
1.3. NOTATION AND DEFINITIONS

1.3.4 Time-delay Embeddings

Consider a time series \( \{x_t\}, t \in \{0, \Delta t, 2\Delta t, \ldots, T-\Delta t, T\} \) in a one-dimensional state space. Since the systems we are interested in are chaotic, they are deterministic and hence each value of the time series is dependent on a finite number of previous values \( x_{t+1} = f(x_t) = f(x_{t-1}) = \cdots \), for some function \( f \). We expect this because within a rapidly sampled time series these values capture essentially the same information as a succession of time derivatives. If we denote the \( n^{th} \) discrete derivative as \( x^n_t = \frac{x^{n-1}_t - x^{n-1}_{t-1}}{(\Delta t)^{n-1}} \), then the first few discrete derivatives of \( \{x_t\} \) are shown below:

\[
x^1_t = \frac{x_t - x_{t-1}}{\Delta t} \quad (1.1)
\]

\[
x^2_t = \frac{x^1_t - x^1_{t-1}}{\Delta t} = \frac{x_t - 2x_{t-1} + x_{t-2}}{(\Delta t)^2} \quad (1.2)
\]

\[
x^3_t = \frac{x^2_t - x^2_{t-1}}{\Delta t} = \frac{x_t - 3x_{t-1} + 3x_{t-2} - x_{t-3}}{(\Delta t)^3} \quad (1.3)
\]

\[
x^4_t = \frac{x^3_t - x^3_{t-1}}{\Delta t} = \frac{x_t - 4x_{t-1} + 6x_{t-2} - 4x_{t-3} + x_{t-4}}{(\Delta t)^4} \quad (1.4)
\]

This dependence on a finite number of previous values for the variable \( x_t \) can be extended to any number of dynamical variables because their past values capture the same information as a succession of partial time derivatives. Dynamical properties such as the eigenvalues of fixed points, largest Lyapunov exponent, and topological properties are contained within (almost) any variable and its time lags, as discussed
1.3. NOTATION AND DEFINITIONS

by Julien Sprott [30]. The time-delay embedding space has the properties that every point in it maps to a unique next point by the dynamics and that there is a smooth nonsingular transformation between it and the original space. The reconstructed embedding space is also Cartesian, rather than curved.

1.3.5 Takens Delay Embedding Theorem

In this thesis, we exploit the result of Takens Delay Embedding Theorem in [30], which states that the embedding lag variables constitute an adequate embedding provided the measured variable is smooth and couples to all the other variables, and the number of embedded lags is at least 2n_s+1. These conditions guarantee that the eigenvalues of fixed points, largest Lyapunov exponent, and topological properties discussed above are preserved in the embedded time series.
Chapter 2

The Poincaré and Higuchi (P&H) Method

Our description of the The Poincaré and Higuchi (P&H) Method is derived from the description in Golestani [9] and Michalak [18]. Our description of the Higuchi Dimension comes from Higuchi [12].

2.1 The Poincaré Section

In general, one can consider a high-dimensional flow in its corresponding phase space or an embedded time series in the same phase space. Often it can be difficult to capture important dynamics of the series in this n-dimensional space. We can, however, observe the dynamics induced by the flow or embedded time series on a particular section of this phase space, defined as the Poincaré Section.

Our phase space is $\mathbb{R}^3$, and we will refer to a Poincaré Section as a Poincaré Plane as we only only consider hyperplanes in $\mathbb{R}^3$. Throughout this thesis, we will denote a Poincaré Plane as $\Sigma \in \mathbb{R}^3$, with normal vector $n \in \mathbb{R}^3$, passing through the point $p \in \mathbb{R}^3$. Without loss of generality, $n$ is always a unit vector, i.e. $\|n\| = 1$, 

16
2.1. THE POINCARÉ SECTION

Figure 2.1:
(a) Intersection method to obtain \( \hat{\Sigma}_{\text{int}} = \{s_1, s_2, s_3, \ldots, s_N\} \).
(b) The Poincaré Map mapping successive intersection points.
Source: [5]

Where \( \| \cdot \| \) is the Euclidean norm. Note that there is not one unique Poincaré Plane of a particular flow or embedded time series.

There are two established methods in which a Poincaré Plane \( \Sigma \) can capture the dynamics of the flow or embedded time series, given in sections 2.1.1 and 2.1.2.

For these methods, we consider the sequence of points \( \Gamma = \{p_t\}_{t=1}^{N_p} \) in \( \mathbb{R}^3 \) given by the flow or embedded map. Note that all flows are defined continuously on their chaotic attractors, but we may obtain a discrete sequence of points by using numerical methods. We use the \textit{ode45} solver in MATLAB, which implements the Runge Kutta method for estimating solutions to ordinary differential equations [14]. In Appendix A the method is outlined in more detail.
2.1. THE POINCARÉ SECTION

2.1.1 Intersection Method (\(\Sigma_{\text{int}}\))

A set of points \(\Sigma_{\text{int}} = \{s_1, s_2, s_3, \ldots, s_{N_s}\}\) is obtained by intersecting \(\Gamma\) with the Poincaré Plane \(\Sigma\) with the condition (Eq. 2.2) that only those intersections are taken into account that occur when the flow goes from one selected side of \(\Sigma\) to the other and not in the opposite direction. This is shown in Figure 2.1(a) and definition of \(\Sigma_{\text{int}}\) given below:

\[
\Sigma_{\text{int}} = \left\{ \overline{p(t)p(t+1)} \cap \Sigma \right\}
\]  

(2.1)

for all \(t\) such that:

\[
\overline{p(t)p(t+1)} \cap \Sigma \neq \emptyset \text{ and } \overline{p(t)p(t+1)} \cdot \overline{n} > 0
\]  

(2.2)

where \(\overline{p(t)p(t+1)}\) denotes the line from point \(p(t)\) to point \(p(t+1)\) and \(\overline{p(t)p(t+1)} \cdot \overline{n}\) denotes the dot product between the vector \(\overline{p(t)p(t+1)}\) and normal of \(\Sigma\). Note also that by convention, the selected side is chosen by \(\overline{p(t)p(t+1)} \cdot \overline{n} > 0\).

2.1.2 Slice Method (\(\Sigma_{\text{slice}}\))

A set of points \(\Sigma_{\text{slice}} = \{s_1, s_2, s_3, \ldots, s_{N_s}\}\) is obtained by projecting the points of \(\Gamma\) onto the Poincaré Plane \(\Sigma\), with the condition (Eq. 2.4) that only the points located near the plane should be taken into consideration. Thus, \(\Sigma_{\text{slice}}\) is given by:

\[
\Sigma_{\text{slice}} = \{p(t) - [(p(t) - p) \cdot \overline{n}] \overline{n}\}
\]  

(2.3)

for all \(t\) such that:

\[
|\overline{(p(t) - p) \cdot \overline{n}}| < \mu
\]  

(2.4)
2.1. THE POINCARÉ SECTION

Figure 2.2:
(a) \( \hat{\Sigma}_{\text{slice}} = \{s_1, s_2, s_3, \ldots, s_N_s\} \) obtained by projecting points from \( \Gamma \) onto the Poincaré Plane \( \Sigma \).
(b) The Poincaré Map applied to each \( s_t \in \hat{\Sigma}_{\text{slice}} \).

Source: [5]

where \( \mu \) is the size of the slice or maximum perpendicular distance from the plane required to project \( p(t) \) onto \( \Sigma \).

In Figure 2.2(a), the set \( \hat{\Sigma}_{\text{slice}} = \{s_1, s_2, s_3, \ldots, s_N_s\} \) is obtained by projecting the points of \( \Gamma \) onto the Poincaré Plane \( \Sigma \), satisfying Eq. 2.4.

2.1.3 Poincaré Map

In sections 2.1.1 and 2.1.2 we captured the dynamics of \( \Gamma \) in \( \hat{\Sigma}_{\text{int}} \subset \Sigma \) and \( \hat{\Sigma}_{\text{slice}} \subset \Sigma \), respectively. The Poincaré Map then connects each successive intersection/projected point on the Poincaré Plane. It is a function \( \mathcal{P} : \Sigma \rightarrow \Sigma \) such that
∀\( t \geq 1 \):
\[
P(s_t) = s_{t+1}
\] (2.5)

The Poincaré Map \( P \) is shown acting on \( \Sigma_{int} \) in Figure 2.1(b). Here, it is a discrete function mapping each point \( s_t \in \hat{\Sigma}_{int} \) to the next successive point: \( P(s_t) = s_{t+1} \).

The Poincaré Map is also shown acting on \( \Sigma_{slice} \) in Figure 2.2(b). Here, it maps each point \( s_t \in \hat{\Sigma}_{slice} \) to the next successive point: \( P(s_t) = s_{t+1} \)

### 2.2 The Higuchi Fractal Dimension

Higuchi [12] proposed a method of calculating the fractal dimension of a time series. We consider the intersection/projected time series on the Poincaré Plane: \( \hat{\Sigma}_{int} = \{s_1, s_2, s_3, \ldots, s_{N_s}\} \) and outline the process similar to Michalak [18].

This method constructs several new time series \( s^k_m \) by starting at \( s_m \) and then selecting every \( k^{th} \) element in the original time series \( \hat{\Sigma}_{int} \). For every value of \( k \) from 1 to a selected maximum time interval \( k_{\text{max}} \), we calculate the average length of the curves containing points spaced every \( k \) samples taken from the original time series. The slope of a linear function between \( \ln(k) \) and the natural logarithm of the length of the curves obtained for each \( k \) is used to calculate the Higuchi dimension.

In greater detail, Higuchi’s method works by performing the following steps:

1. A set \( \mathcal{K} \) of time intervals is chosen. In the original work by Higuchi [12], the union of \( k = 1, 2, 3, 4 \) and \( k = \lfloor 2^{\left\lfloor \frac{j-1}{4} \right\rfloor} \rfloor \) was used, where \( j = 11, 12, 13, \ldots \) and \( \lfloor \cdot \rfloor \) denotes the floor function. A maximum value for \( k \) was set to \( k_{\text{max}} = 2^{11} \). Thus, the set \( \mathcal{K} \) used by Higuchi was:
2.2. THE HIGUCHI FRACTAL DIMENSION

\[ \mathcal{K} = \{1, 2, 3, 4\} \cup \left\{ \left\lfloor 2^{\left(\frac{j-1}{4}\right)} \right\rfloor \right\}_{j=11,12,13,...,45} \]  \hspace{1cm} (2.6)

In Golestani et al. [9], \( k_{\text{max}} = 20 \) was chosen and thus the set:

\[ \mathcal{K} = \{1, 2, 3, \ldots, 20\} \]  \hspace{1cm} (2.7)

Since we are also applying Higuchi’s fractal dimension within the P&H method, we use the set above (Eq. 2.7).

2. \( \forall k \in \mathcal{K} \), we fix the time interval \( k \), and \( \forall m \in \{1, 2, 3, \ldots, k\} \), we construct new time series \( s^k_m \) by taking elements from the original time series \( s_t \) in the following manner:

\[ s^k_m = \left\{ s_m, s_{m+k}, s_{m+2k}, \ldots, s_{m+\left\lfloor \frac{N-m}{k} \right\rfloor \cdot k} \right\} \]  \hspace{1cm} (2.8)

Thus, we obtain \( k \) new time series from the original one, each constructed with a different starting point. For example, if we had the time series \( \{s_1, s_2, s_3, \ldots, s_{100}\} \), when we fix \( k = 3 \), the following 3 time series are obtained:

\[ s^3_1 = \{s_1, s_4, s_7, \ldots, s_{100}\} \]

\[ s^3_2 = \{s_2, s_5, s_8, \ldots, s_{98}\} \]  \hspace{1cm} (2.9)

\[ s^3_3 = \{s_3, s_6, s_9, \ldots, s_{99}\} \]

Note that after obtaining \( k \) new time series for each fixed \( k \in \mathcal{K} \), in total we should obtain:

\[ \sum_{k=1}^{k_{\text{max}}} k = \frac{(k_{\text{max}})(k_{\text{max}} + 1)}{2} \]  \hspace{1cm} (2.10)

new \( s^k_m \) time series over the set \( \mathcal{K} \). In our experiments, since \( k_{\text{max}} = 20 \), in total
we obtain \( \frac{(20)(21)}{2} = 210 \) new \( s^k_m \) time series over the set \( K \).

3. For each of the new time series \( s^k_1, s^k_2, s^k_3, \ldots, s^k_k \) belonging to the time
   interval \( k \), the length \( L_m(k) \) of the curve connecting the elements of \( s^k_m \) is calcu-
   lated:

\[
L_m(k) = \frac{\sum_{i=1}^{\lfloor \frac{N-m}{k} \rfloor} |s_{m+ik} - s_{m+(i-1)k}| \left( \frac{N-m}{\lfloor \frac{N-m}{k} \rfloor} \right)}{k} \quad (2.11)
\]

In Eq. 2.11 above, the term \( \sum_{i=1}^{\lfloor \frac{N-m}{k} \rfloor} |s_{m+ik} - s_{m+(i-1)k}| \) simply sums all of the
Euclidean distances between successive points of the Poincaré Map, and the term
\( \frac{N-m}{\lfloor \frac{N-m}{k} \rfloor} \) is a normalization factor given by Higuchi.

4. The average length \( L(k) \) is calculated for each time interval \( k \) using the
   lengths \( L_m(k) \), \( m = 1, 2, 3, \ldots, k \):

\[
L(k) = \frac{\sum_{m=1}^{k} L_m(k)}{k} \quad (2.12)
\]

5. The Higuchi fractal dimension is determined as such value \( D \) that satisfies
   the following porportionality:

\[
L(k) \propto k^{-D_H} \Rightarrow D_H = -\frac{\ln(L(k))}{\ln(k)} \quad (2.13)
\]

Thus, when \( \ln(L(k)) \) is plotted against \( \ln(k) \) the points should fall on a line with
a slope of \(-D\).
2.3. THE P&H METHOD

The P&H method uses the Poincaré Section (section 2.1) and applies most of the steps involved in the calculation of the Higuchi dimension (section 2.2). There are two important modifications with respect to the procedure used by Higuchi.

1. First, as implemented by both Golestani [9] and Michalak [18], the P&H method uses the normalization factor of:

\[ N - m \left\lfloor \frac{N - m}{k} \right\rfloor k \]  

(2.14)

instead of:

\[ \frac{N - m}{\left\lfloor \frac{N - m}{k} \right\rfloor} \]  

(2.15)

as used in Eq. 2.11

2. Second, in the P&H method we do not actually calculate the fractal dimension, so no line fitting is performed for \( \ln(L(k)) \). Instead, a graph of \( \ln(L(k)) \)
2.3. THE P&H METHOD

Figure 2.4: The smooth decreasing trend of the P&H method’s outputs for stochastic time series. Source: [9]

versus $k$ is plotted. The characteristics of this graph can be used for determining if the time series was generated by a deterministic chaotic process or by a stochastic one. In the former case the pattern visible in the graph is jagged and has frequent, pronounced increases and decreases, as shown in Figure 2.3. In the latter case a more smooth, decreasing curve can be observed, as shown in Figure 2.4.
Chapter 3

Time Series Data Sets

Because we are building upon the optimization methods of Krzysztof Michalak [18], we use the same twelve time series in our experiments: six generated by deterministic chaotic systems and six generated by stochastic processes.

All stochastic and chaotic 1-dimensional time series were embedded into $\mathbb{R}^3$, with a lag of $\tau = 1$, before applying the P&H method. All time series, once embedded into $\mathbb{R}^3$ if necessary, form a sequence of points $\Gamma = \{p_i\}_{i=1}^{N_p} \subset \mathbb{R}^3$.

3.1 Stochastic Time Series

The six data sets used in the experiments generated by stochastic processes were: Gaussian 1D, Gaussian 3D, Random Walk 1D, Random Walk 3D, Uniform 1D and Uniform 3D. They were generated by a program provided by Krzysztof Michalak [17], which he used to generate the six stochastic time series in Michalak [18].

3.1.1 Gaussian (Normal) 1D

This data set is a one-dimensional time series with all the elements independently drawn from a one-dimensional normal distribution $\mathcal{N}(0, 1)$ with a mean $\mu = 0$.
and a standard deviation of $\sigma = 1$.

3.1.2 Gaussian 3D

This data set is a three-dimensional time series with all the elements independently drawn from a three-dimensional normal distribution with a mean $\mu = (0, 0, 0)$ and an identity covariance matrix $\mathcal{C}OV = I^3$.

3.1.3 Random Walk 1D

This data set contains a one-dimensional time series with the first value equal to 0 and each consecutive element obtained by adding a value drawn from a one-dimensional normal distribution $\mathcal{N}(0, 1)$ with a mean $\mu = 0$ and a standard deviation of $\sigma = 1$ to the previous one.

3.1.4 Random Walk 3D

This data set contains a three-dimensional time series with the first value $p_0 = (0, 0, 0)$ and each consecutive element obtained by adding a three-dimensional vector drawn from a three-dimensional normal distribution with a mean $\mu = (0, 0, 0)$ and an identity covariance matrix $\mathcal{C}OV = I^3$.

3.1.5 Uniform 1D

This data set is a one-dimensional time series with all the elements independently drawn from a one-dimensional uniform distribution $\mathcal{U}(0, 1)$. 
3.2 Deterministic Time Series

The data sets used in the experiments generated by deterministic chaotic systems were: Logistic, Bouali, ChenLee, Lorenz, RabinovichFabrikant and Rössler.

Note that the latter five of these time series (Bouali, ChenLee, Lorenz, Rabinovich-Fabrikant and Rössler) are 3-dimensional chaotic flows. This means that they are defined by three differential equations in three variables, producing continuous trajectories on their respective chaotic attractor. For our analysis of these flows, we require a discrete sequence of points, which is a subset of the trajectory for a defined time step.

The Bouali, ChenLee, Lorenz, and Rössler systems were produced in Matlab using the *ode45 solver*, an implementation of the fourth order Runge-Kutta method [14] (see Appendix A for the Runge-Kutta method). The RabinovichFabrikant and Rössler systems were downloaded as discrete time series from Sprott [32]. The Logistic map was also downloaded as discrete time series from Sprott [31].
3.2. DETERMINISTIC TIME SERIES

3.2.1 Logistic map

This time series is generated by a discrete-time system and was introduced in 1962 by Myrberg [22] to model population growth. It is given by the equation:

\[ x_{n+1} = rx_n(1 - x_n) \]  \hspace{1cm} (3.1)

In the experiments, parameters \( r = 3.8284 \) and \( x_0 = 0.712765 \) were used as in Michalak [18]. Figure 3.2 shows the bifurcation diagram for the Logistic map. For each fixed value of the parameter \( r \) on the x-axis, the forward limit set (Section 1.3.22) is approximated. A bifurcation refers to a period doubling, quadrupling, etc., that accompanies the onset of chaos. [27] At each point where there is a branching of the plot in Figure 3.2, a bifurcation occurs. \( r = 3.8284 \) is an appropriate parameter value for analysis of chaos in this map as it approximately contains the most dense

Figure 3.1: Collection of attractors for embeddings of the Logistic map into \( \mathbb{R}^3 \), with lag \( \tau = 1 \) and parameter \( r \in [3.6, 4.0] \).
Source: [3]
3.2. DETERMINISTIC TIME SERIES

Figure 3.2: Logistic map bifurcation diagram for the parameter $r \in [2.4, 4.0]$. Source: [21]

forward limit set before a period-3 doubling occurs. [27]

Figure 3.3: Logistic map embedded into $\mathbb{R}^3$ with lag $\tau = 1$ and parameters $r = 3.8284$ and $x_0 = 0.712765$. Lines between successive points in the embedded map are shown, which intersect the Poincaré Plane.
3.2. DETERMINISTIC TIME SERIES

3.2.2 Bouali

A system introduced by Bouali [4] in 2012. This system consists of three ordinary differential equations:

\[
\begin{align*}
\frac{dx}{dt} &= x(a - y) + \alpha z \quad (3.2) \\
\frac{dy}{dt} &= -y(b - x^2) \quad (3.3) \\
\frac{dz}{dt} &= -x(c - sz) - \beta z \quad (3.4)
\end{align*}
\]

Parameters of this system were \(a = 4\), \(b = 1\), \(c = 1.5\), \(s = 1\), \(\alpha = 0.3\), \(\beta = 0.05\), and initial condition \((x_0, y_0, z_0) = (0.5, 0.5, 0.5)\). Its chaotic attractor and the time series data used for our analysis are shown in Figure 3.4.
3.2. DETERMINISTIC TIME SERIES

Figure 3.4:
(a) Bouali system chaotic attractor; parameters: $a = 4$, $b = 1$, $c = 1.5$, $s = 1$, $\alpha = 0.3$, $\beta = 0.05$, (noise) $\eta = 0$; initial condition $(x_0, y_0, z_0) = (0.5, 0.5, 0.5)$.
(b) 1500 point time series generated from Bouali system above. Produced in Matlab (ode45-Runge Kutta).
3.2.3 Chen-Lee

Proposed in 2004 by Chen and Lee in a paper concerning the behavior of a rigid body influenced by certain feedback gains. This system consists of the equations:

\[
\begin{align*}
\frac{dx}{dt} &= -yz + ax \\
\frac{dy}{dt} &= xz + by \\
\frac{dz}{dt} &= \frac{1}{3}xy + cz
\end{align*}
\]  

Parameters of this system were \( a = 5 \), \( b = 10 \), \( c = 3.8 \), and initial condition \( (x_0, y_0, z_0) = (1, 1, 1) \). Its chaotic attractor and the time series data used for our analysis are shown in Figure 3.5.
3.2. DETERMINISTIC TIME SERIES

Figure 3.5:
(a) Chen-Lee system chaotic attractor; parameters: $a = 5$, $b = -10$, $c = -3.8$, (noise) $\eta = 0$; initial condition $(x_0, y_0, z_0) = (1, 1, 1)$.
(b) 1500 point time series generated from Chen-Lee system above. Produced in Matlab (ode45-Runge Kutta).
3.2. DETERMINISTIC TIME SERIES

3.2.4 Lorenz

A system developed in 1963 by Edward Lorenz as a simplified mathematical model for atmospheric convection. This system consists of three ordinary differential equations:

\[
\frac{dx}{dt} = \sigma(y - x) \quad (3.8)
\]

\[
\frac{dy}{dt} = x(\rho - z) - y \quad (3.9)
\]

\[
\frac{dz}{dt} = xy - \beta z \quad (3.10)
\]

Parameters of this system were $\sigma = 10$, $\rho = 28$, $\beta = \frac{8}{3}$, and initial condition $(x_0, y_0, z_0) = (0.5, 0.5, 0.5)$. Its chaotic attractor and the time series data used for our analysis are shown in Figure 3.6.
3.2. DETERMINISTIC TIME SERIES

Figure 3.6:
(a) Lorenz system chaotic attractor; parameters: $\sigma = 10$, $\rho = 28$, $\beta = \frac{8}{3}$, (noise) $\eta = 0$; initial condition $(x_0, y_0, z_0) = (0.5, 0.5, 0.5)$.
(b) 1500 point time series generated from Lorenz system above. Produced in Matlab (ode45-Runge Kutta).
3.2. DETERMINISTIC TIME SERIES

3.2.5 Rabinovich-Fabrikant

A system introduced by Rabinovich and Fabrikant \[26\] in 1979 for modeling the dynamical behavior caused by the modulation instability in a non-equilibrium dissipative medium. The RF system is an example of a dynamical system with multiple attractors. This system consists of three ordinary differential equations:

\[
\begin{align*}
\frac{dx}{dt} &= y(z - 1 + x^2)\gamma x \\
\frac{dy}{dt} &= x(3z + 1 - x^2)\gamma y \\
\frac{dz}{dt} &= -2z(\alpha + xy)
\end{align*}
\]

Parameters of this system were $\alpha = 1.1$, $\gamma = 0.87$, and initial condition $(x_0, y_0, z_0) = (-1, 0, 0.5)$.

Like all chaotic systems, this system is very sensitive to the initial conditions and time steps used when trying to produce a proper data set on the chaotic attractor. After exploring various initial conditions and settings put forth by Danca et al. \[6\] and Julien C. Sprott \[32\], we were able to produce its chaotic attractor and the time series data used for our analysis, as shown in Figure 3.6.
3.2. DETERMINISTIC TIME SERIES

Figure 3.7:
(a) Rabinovich-Fabrikant system chaotic attractor; parameters: $\sigma = 10$, $\rho = 28$, $\beta = \frac{8}{3}$, (noise) $\eta = 0$; initial condition $(x_0, y_0, z_0) = (-1, 0, 0.5)$.

(b) 1500 point time series generated from Rabinovich-Fabrikant system above. Produced in Matlab (ode45-Runge Kutta).
3.2. DETERMINISTIC TIME SERIES

3.2.6 Rössler

Probably the most popular example of the simple folding was proposed in 1976 by Rössler[33]. This system consists of three ordinary differential equations:

\[
\begin{align*}
\frac{dx}{dt} &= -y - z \\
\frac{dy}{dt} &= x + ay \\
\frac{dz}{dt} &= b + z(x - c)
\end{align*}
\]

Parameters of this system were \( a = 0.2, b = 0.2, c = 5.7 \), and initial condition \((x_0, y_0, z_0) = (1, 1, 1)\). Its chaotic attractor and the time series data used for our analysis are shown in Figure 3.8.
3.2. DETERMINISTIC TIME SERIES

Figure 3.8:

(a) Rössler system chaotic attractor; parameters: \( a = 0.2, b = 0.2, \) and \( c = 5.7, \) (noise) \( \eta = 0; \) initial condition \( (x_0, y_0, z_0) = (1, 1, 1). \)

(b) 1500 point time series generated from Rössler system above. Produced in Matlab (ode45-Runge Kutta).
3.3 Adding Gaussian White Noise to Data Sets

We study the influence of noise on the possibility of detecting deterministic behavior. Thus, Gaussian White Noise was added to all chaotic data sets. The parameter controlling the level of noise in our experiments is $\eta \in [0, 0.1]$, analogous to Golestani et al. [9] and Michalak [18]. For each coordinate $x$, $y$, and $z$, the standard deviation of points in the data set was calculated: $\sigma_x$, $\sigma_y$, and $\sigma_z$, respectively. We first generate three independent Gaussian random variables:

\[
\hat{x} \sim N(0, \eta \sigma_x) \\
\hat{y} \sim N(0, \eta \sigma_y) \\
\hat{z} \sim N(0, \eta \sigma_z)
\]

(3.17)

Then, for each point of the flow $\Gamma = \{p_t\}_{t=1}^{N_p}$ in $\mathbb{R}^3$, the random vector generated above is added:

\[
(x', y', z') = (x, y, z) + (\hat{x}, \hat{y}, \hat{z})
\]

(3.18)

For $\eta = 0$, no noise was added, and an increasing level of noise was introduced to each data set, with a maximum of $\eta = 0.1$ noise. For the Logistic map, noise was added using the same process with only one coordinate.
Chapter 4

Optimization of the Poincaré Plane in $\mathbb{R}^3$

It is important to note that the present formulation of the optimization problem was given by Krzysztof Michalak [18]. In Section 4.1 we outline the optimization problem with Michalak’s objective function and in Section 4.2 we propose our own.

Throughout this section, we will denote the Poincaré plane as $\Sigma \in \mathbb{R}^3$ with normal vector $n$ and passing through the point $p$. Note also that $\ln |L(k)|$ are the logarithmic lengths calculated in Section 2.2 $\ln |L(k)|' = \ln |L(k+1)| - \ln |L(k)|$ and $\ln |L(k)|'' = \ln |L(k+1)|' - \ln |L(k)|'$ are backward first and second discrete derivatives, respectively.

4.1 Current Objective Function

Michalak [18] chose the objective function $\Delta$ (Eq. 4.1) because out of the several evaluation criteria he applied, it “produced the most distinctive zigzag patterns for deterministic dynamical systems” [18] in the P&H graph.

For all algorithms, if using the intersection method, we solve the following un-
4.2. NEW OBJECTIVE FUNCTION

constrained optimization problem:

\[
\max \Delta(p, n) = \sum_{k=1}^{k_{\text{max}}-3} (\ln |L(k+1)|''' - \ln |L(k)|''')^2
\] (4.1)

For all algorithms, if using the slice method, we must also include the slice parameter \(\mu\). Thus we solve the following unconstrained optimization problem:

\[
\max \Delta_{v_s}(p, n, \mu) = \sum_{k=1}^{k_{\text{max}}-3} (\ln |L(k+1)|''' - \ln |L(k)|''')^2
\] (4.2)

We have implemented and used both the intersect and the slice methods, but without loss of generality, we usually refer to Eq. 4.1 as the objective function.

4.2 New Objective Function

We present a new objective function, which is able to capture distinctive zigzag patterns for deterministic dynamical systems in the P&H graph, as did \(\Delta\) (Eq. 4.1).

For all algorithms, if using the intersection method, we solve the following unconstrained optimization problem:

\[
\max \psi(p, n) = \sum_{k=1}^{k_{\text{max}}-3} |\ln |L(k+1)||' - \ln |L(k)||'| \] (4.3)

For all algorithms, if using the slice method, we must also include the slice parameter \(\mu\). Thus we solve the following unconstrained optimization problem:

\[
\max \psi_{v_s}(p, n, \mu) = \sum_{k=1}^{k_{\text{max}}-3} |\ln |L(k+1)||' - \ln |L(k)||'| \] (4.4)
4.3 Summary of P&H Method and Objective Function

Figure 4.1 summarizes the P&H Method and optimization of the function $\Delta$ (Eq. 4.1). Starting at the upper left, if we have a 1-dimensional time series $\{x_t\}_{t=1}^{N_x} \subset \mathbb{R}$, we embed it into $\Gamma = \{p_t\}_{t=1}^{N_p} \subset \mathbb{R}^3$ with the Embedding function $E_1 : \mathbb{R} \rightarrow \mathbb{R}^3$. Note that the subscript in $E_1$ refers to an embedding with a lag of $\tau = 1$.

Next, we optimize over the parameters $(p, n) \in \mathbb{R}^3 \times \mathbb{R}^3$ found at the top of Figure 4.1. The function $\beta : \mathbb{R}^3 \times \mathbb{R}^3 \rightarrow \Sigma$ defines the Poincaré plane $\Sigma$ with normal vector $n$, passing through the point $p$.

Starting at the left of the bottom row, we first take the intersection of $\Gamma \subset \mathbb{R}^3$ and the Poincaré plane $\Sigma$. The Poincaré Map $\mathcal{P} : \Sigma \rightarrow \Sigma$ then connects successive intersection points on the Poincaré plane.

The modified Higuchi function $\tilde{\mathcal{H}} : \Sigma \rightarrow \mathbb{R}^{k_{max}}$ produces a vector of logarithmic lengths $(\ln |L(1)|, \ln |L(2)|, \ldots, \ln |L(k_{max})|)$.

Finally, $\tilde{\Delta} : \mathbb{R}^{k_{max}} \rightarrow \mathbb{R}$ calculates the square of the backward third discrete
4.3. SUMMARY OF P&H METHOD AND OBJECTIVE FUNCTION

derivatives to give the result in Eq. 4.1

\[
\Delta(p, n) = \sum_{k=1}^{k_{\text{max}}-3} (\ln |L(k+1)|'' - \ln |L(k)|'')^2.
\]
Chapter 5

Previous Optimization Methods

The following methods to optimize the Poincaré Plane (Coordinate Axes, PCAmin, PCAmax, Gradient, and Evolutionary Algorithm) were proposed by Krzysztof Michalak [18] to solve the optimization problems in Chapter 4. Their implementation is available at [19].

5.1 Coordinate Axes

This algorithm fixes $p = m$, where $m$ is the mean of the flow/embedded map sequence $\{p_t\}_{t=1}^{N_p}$. It then chooses the normal from the coordinate axes ($n \in \{e_1, e_2, e_3\}$) that optimizes:

$$\max \Delta(p, n) = \sum_{i=1}^{k_{max}-3} (y''_{i+1} - y''_i)^2$$  \hspace{1cm} (5.1)

5.2 Principle Component Analysis (PCAmim/PCAmax)

Fix $p = m$, where $m$ is the mean of the flow/embedded map sequence $\Gamma = \{p_t\}_{t=1}^{N_p} \subset \mathbb{R}^3$. 

45
5.2. PRINCIPLE COMPONENT ANALYSIS (PCAMIN/PCAMAX)

We then calculate the covariance matrix for $\Gamma$. The covariance matrix is a symmetrical matrix of covariance between the variables of our embedding space (or configuration space if a dynamic flow) $\mathbb{R}^3$. By convention we use the variables $x$, $y$, and $z$ and the covariance between two of these variables is defined as:

$$cov_{x,y} = \frac{\sum_{i=1}^{N} (x_i - \bar{x})(y_i - \bar{y})}{N_p - 1}$$

(5.2)

where $\bar{x}$ is the mean of the variable $x$ and $N_p$ is the number of points in the flow/embedded map sequence. We end up with the symmetric $3 \times 3$ covariance matrix:

$$\text{COV} = \begin{bmatrix}
    cov_{x,x} & cov_{x,y} & cov_{x,z} \\
    cov_{y,x} & cov_{y,y} & cov_{y,z} \\
    cov_{z,x} & cov_{z,y} & cov_{z,z}
\end{bmatrix}$$

(5.3)

$\text{COV}$ contains the linear relationship between variables and the PCA method attempts to capture this through the eigenvectors of $\text{COV}$, as shown for points in $\mathbb{R}^2$ in Figure 5.1.

Next we find the eigenvectors and corresponding eigenvalues of $\text{COV}$:

$$\text{COV} \begin{bmatrix} \vec{v}_1 & \vec{v}_2 & \vec{v}_3 \end{bmatrix} = \begin{bmatrix} \vec{v}_1 & \vec{v}_2 & \vec{v}_3 \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 & 0 \\
0 & \lambda_2 & 0 \\
0 & 0 & \lambda_3 \end{bmatrix}$$

(5.4)

The eigenvectors give a change of basis for the data in terms of ranking covariance. For PCAmx we assign $n = \vec{v}_1$, the eigenvector with the highest ranking eigenvalue, and for PCAmn we assign $n = \vec{v}_3$, the eigenvector with the lowest ranking eigenvalue.
5.2. PRINCIPLE COMPONENT ANALYSIS (PCAMIN/PCAMAX)

Figure 5.1: A basis obtained using the Principal Component Analysis (PCA) method for a two-dimensional set of points.
Source: [18]
5.3. Gradient

This method uses the gradient descent optimization method, which iteratively converges to an optimum using the function gradient to direct the search. This algorithm considers the parameters of the Poincaré Plane as a single vector $v \in \mathbb{R}^6$. In the case of the slice method, we consider a single vector $v \in \mathbb{R}^6$, where the last coordinate is the slice parameter $\mu$.

As shown in Figure 5.2, the optimization is performed iteratively starting from an initial point $v_0 \in \mathbb{R}^6$, chosen randomly in the solution space.

Due to the discrete nature of both objective functions, in each step the algorithm must estimate the gradient of the function (black arrow) at the currently known maximum point (black dot). It then searches for the next point $v \in \mathbb{R}^6$ along the
5.4. EVOLUTIONARY ALGORITHM

Figure 5.3: A genotype used in Krzysztof Michalak’s evolutionary algorithm with the slice method. Source: [18]

direction of the gradient that maximizes the objective function.

To avoid being trapped in local maxima, the method allows for restarts of the algorithm. The default number of restarts is 10.

5.4 Evolutionary Algorithm

This heuristic algorithm, implemented by Krzysztof Michalak [18] processes a population of solutions which undergo the processes of selection, crossover and mutation. It uses the mechanism of elitism: it promotes the best specimen from the current generation and its mutated copy to the next generation without the need of winning in the selection process.

Similar to the Gradient method above, it also considers the parameters of the Poincaré Plane as a single vector \( v \in \mathbb{R}^6 \). In the case of the slice method, we consider a single vector \( v \in \mathbb{R}^6 \), where the last coordinate is the slice parameter \( \mu \). Evolutionary algorithms attempt to parallel biological evolution and hence vectors of the solution space are termed “specimens” and the coordinates of these vectors constitute their “genes”.

The flow map of the algorithm is given in Figure 5.4 and explained in detail below:
5.4. EVOLUTIONARY ALGORITHM

IN:
- \( N_{\text{gen}} \) - the number of generations
- \( N_{\text{pop}} \) - the size of the population
- \( P_{\text{cross}} \) - crossover probability
- \( P_{\text{mut}} \) - mutation probability

OUT:
- \( S^* = \emptyset \) - the best solution found by the algorithm

\[
P = \text{InitPopulation}(N_{\text{pop}})
\]

\[
\text{for } g = 1, \ldots, N_{\text{gen}} \text{ do}
\]
\[
\text{Evaluate}(P)
\]
\[
S_0 = \text{SelectBestSpecimen}(P)
\]
\[
S_1 = \text{Mutate}(S_0)
\]
\[
P' = \text{SelectMatingPool}(P, N_{\text{pop}} - 2)
\]
\[
P'' = \text{Crossover}(P', P_{\text{cross}})
\]
\[
\text{for } i = 1, \ldots, N_{\text{pop}} - 2 \text{ do}
\]
\[
\text{if } \text{Rand}(U[0, 1]) < P_{\text{mut}} \text{ then}
\]
\[
\text{Mutate}(P''[i])
\]
\[
\text{end if}
\]
\[
\text{end for}
\]
\[
P = P'' \cup \{S_0\} \cup \{S_1\}
\]
\[
S^* = \{ \text{SelectBestSpecimen}(P \cup S^*) \}
\]
\[
\text{end for}
\]

Figure 5.4: An overview of the evolutionary algorithm Krzysztof Michalak implemented.
Source: [18]
5.4. EVOLUTIONARY ALGORITHM

**InitPopulation** - Initiates the population $P$. The first 3 coordinates of a new solution (point $p$) are initialized to random values drawn from a normal distribution with the mean and variance equal to those estimated from the flow/embedded map $\Gamma$. The next 3 coordinates (normal $n$) are initialized to random values drawn from a normal distribution $\mathcal{N}(0, 0.1)$. The “slice” gene $\eta$ is initialized with a random variable drawn from the normal distribution $\mathcal{N}(0, 0.1)$.

**Evaluate** - For each specimen in the population $P$ this procedure calculates the value of $\Delta_v$ (Eq. 4.1) for intersect the method and $\Delta_{vs}$ (Eq. 4.2) for the slice method.

**SelectBestSpecimen** - Selects the specimen with the highest evaluation value from the current population $P$.

**SelectMatingPool** - Selects $(N_{pop} - 2)$ specimens into a mating pool $P'$ with probability of selecting a specimen proportional to the evaluation of this specimen (to mate with).

**Crossover** - Produces a new population $P''$ based on a given mating pool $P'$. For each consecutive pair of (mating) specimens from the mating pool, a new pair of specimens is created with the probability $P_{cross}$, and with the probability $1 - P_{cross}$ both specimens are copied to the next generation without changes. In this paper uniform crossover is used, which for every position in the genotype copies the value from one of the two parent specimens into one of the two offspring with $1/2$ probability. Thus, this operator copies approximately one half of the values from one parent and one half from the other parent into each new offspring.

**Mutation** - This operator selects one position in the genotype with uniform probability and then adds a value drawn from a normal distribution $\mathcal{N}(0, 0.1)$ to the value
of the selected coordinate in the genotype.

After $N_{\text{gen}}$ (default $N_{\text{gen}} = 50$) generations are completed, the genotype of the specimen with the highest evaluation found among all the specimens processed is returned as the best solution found by the algorithm.
Chapter 6

Analysis of the Solution Space

As described in Chapter 4, to optimize the placement of the Poincaré Plane in $\mathbb{R}^3$, our search space can be seen as either $\mathbb{R}^3 \times \mathbb{R}^3$ or $\mathbb{R}^6$ for the intersection method and as $\mathbb{R}^3 \times \mathbb{R}^3 \times \mathbb{R}$ or $\mathbb{R}^7$ for the slice method. We consider the intersection method and attempt to find new optimization strategies locally within the search space.

This analysis of the search space led to the two new methods of optimization: Random Sample (Section 7.1) and Local Sphere Cap Search (Section 7.2).

6.1 Sphere Gradient Map

This analysis of the search space is termed Sphere Gradient Map because it produces a gradient map of the objective function value (Eq. 4.1) on the unit sphere centred at a fixed point in $\mathbb{R}^3$.

In detail, the procedure is as follows:

1. Fix a specific $p$ as the point on the Poincaré Plane $\Sigma$.
2. Generate an arbitrarily large number ($N_{sp}$) of sample points uniformly on
the unit sphere (see Appendix C) as possible normals for $\Sigma$, namely:

$$\mathcal{S} = \{ n_i : i = 1, 2, 3, \ldots, N_{sp}, n_i \in \mathbb{R}^3 \text{ and } |n_i| = 1 \}$$

3. For all $n_i \in \mathcal{S}$, we compute the objective function $\Delta(p, n_i)$ given in Eq. 4.1.

A gradient map based on the objective value $\Delta$ is created.

A collection of sphere gradient maps are shown in Figures 6.1 - 6.8. We learned from this analysis that for many points in the flow/embedded time series, there are well-defined regions with high values of the objective function $\Delta$. Many of these well-defined regions resemble a cap of the unit sphere, especially in Figure 6.4 and Figure 6.8. This realization led to the Local Sphere Cap Search in Section 7.2, which samples caps of the unit sphere centred at each point of the flow/embedded time series.
6.1. SPHERE GRADIENT MAP

Figure 6.1: Sphere Gradient Map for the 1-D Gaussian Stochastic time series centred at the fixed point $p = (0, 0, 0)$. Points are the embedded time series and scale on right is the value of the objective function $\Delta$ (Eq. 4.1)

Figure 6.2: **Zoom Figure 6.1** Sphere Gradient Map for the 3-D Uniform Stochastic time series centred at the fixed point $p = (0, 0, 0)$. Scale on right is the value of the objective function $\Delta$ (Eq. 4.1)
6.1. SPHERE GRADIENT MAP

Figure 6.3: Sphere Gradient Map for the 1-D Guassian Stochastic time series centred at the fixed point \( p = (0, 0, 0) \). Points are the embedded time series and scale on right is the value of the objective function \( \Delta \) (Eq. 4.1)

Figure 6.4: Zoom Figure 6.3 Sphere Gradient Map for the 3-D Uniform Stochastic time series centred at the fixed point \( p = (0, 0, 0) \). Scale on right is the value of the objective function \( \Delta \) (Eq. 4.1)
6.1. SPHERE GRADIENT MAP

Figure 6.5: Sphere Gradient Map for the embedded Henon Chaotic Map centred at the fixed point \( p = (0.93705422, 0.133270569, 0.477020105) \). Points are the embedded time series and scale on right is the value of the objective function \( \Delta \) (Eq. 4.1).
6.1. SPHERE GRADIENT MAP

Figure 6.7: Sphere Gradient Map for the embedded Logistic Chaotic Map centred at the fixed point \( p = (0.7689, 0.6213, 0.8835) \) with noise \( \eta = 0.1 \). Points are the embedded time series and scale on right is the value of the objective function \( \Delta \) (Eq. 4.1).

Figure 6.8: Zoom Figure 6.7 Sphere Gradient Map for the embedded Logistic Chaotic Map centred at the fixed point \( p = (0.7689, 0.6213, 0.8835) \) with noise \( \eta = 0.1 \). Scale on right is the value of the objective function \( \Delta \) (Eq. 4.1).
6.2 Flow Gradient Map

This analysis of the search space is termed the Flow Gradient Map because it produces a gradient map of the objective function value (Eq. 4.1) for each point of the flow/embedded time series sequence $\Gamma$ in $\mathbb{R}^3$.

The procedure uses the flow/embedded map sequence $\Gamma = \{p_t\}_{t=1}^{N_p}$ as the candidates for the point $p$ of the Poincaré Plane $\Sigma$. Thus for all $t \in \{1, \ldots, N_p\}$:

1. Fix each point $p = p_t$ as the point on the Poincaré Plane $\Sigma$.

2. Generate an arbitrarily large number ($N_{sp}$) of sample points uniformly on the unit sphere (see Appendix C) as possible normals for $\Sigma$, namely:

$$\mathcal{S} = \{n_i : i = 1, 2, 3, \ldots, N_{sp}, n_i \in \mathbb{R}^3 \text{ and } |n_i| = 1\}$$

Figure 6.9: Sphere Gradient Map for Logistic Map, with no noise. Scale on right is the value of the objective function $\Delta$ (Eq. 4.1)
3. For all \( n_i \in S \), we compute the objective function \( \Delta(p, n_i) \) given in Eq. 4.1. Assign the maximum objective function value (\( p_{\Delta_{\text{max}}} \)) as the gradient score of the fixed point \( p \):

\[
p_{\Delta_{\text{max}}} = \max_{n_i} \Delta(p_t, n_i)
\]

After steps 1 - 3 have been applied for all \( t \in \{1, \ldots, N_p\} \), each point in the flow/embedded time series sequence \( \Gamma \) has been assigned a gradient score (\( p_{\Delta_{\text{max}}} \)), which creates a gradient map, as shown in Figures 6.9 and 6.10.

---

Figure 6.10: Sphere Gradient Map for Logistic Map, with noise \( \eta = 0.1 \). Scale on right is the value of the objective function \( \Delta \) (Eq. 4.1).
6.3 Normal Gradient Map

This analysis of the search space is termed the *Normal Gradient Map* because it produces a gradient map of the objective function value (Eq. 4.1) for the translation of a fixed Poincaré Plane along its normal.

In detail, the procedure is as follows:

1. Fix a specific $p$ and $n$ as the point and normal, respectively, of the Poincaré Plane $\Sigma$. $p$ and $n$ also define a line, $\vec{p}(t) = nt + p$, $t \in \mathbb{R}$ upon which we translate the plane in the direction of its normal.

2. Sample an arbitrary number ($N_{sp}$) of sample points uniformly on $(-t_{\text{max}}, t_{\text{max}})$, to generate possible points for $\Sigma$ along a portion of the line $\vec{p}(t)$, namely:

$$S = \{p_{t_i} = nt_i + p : t_i \sim \mathcal{N}(-t_{\text{max}}, t_{\text{max}}), 1 \leq i \leq N_{sp}\}$$

![Sphere Gradient Map of Squared Variation (Logistic Chaotic Time Series; noise $\eta = 0.1$)](image)

Figure 6.11: Normal Gradient Map Gradient for Logistic Map, with noise $\eta = 0.1$. Scale on right is the value of the objective function $\Delta$ (Eq. 4.1)
3. For all $p_t \in S$, we compute the objective function $\Delta(p_t, n)$ given in Eq. 4.1.

A gradient map based on the objective value $\Delta$ is created on the line $\vec{p}(t)$. This is essentially the fixed plane translated along its normal in both directions, as shown in Figures 6.11 and 6.12.

As shown in Figures 6.11 and 6.12, clearly the further the plane is from the points of $\Gamma$, the lower the objective score. This reinforced the our assumption that the candidates for the point $p$ on the plane should be the points from the flow/embedded time series $\Gamma$.

Figure 6.12: **Zoom**: Normal Gradient Map Gradient for Logistic Map, with noise $\eta = 0.1$. Scale on right is the value of the objective function $\Delta$ (Eq. 4.1).
Chapter 7

New Optimization Methods

The following optimization methods (Random Sample and Local Sphere Cap Search) are heuristic methods proposed by this thesis to solve the problem in Chapter 4. They are implemented in the C++ program OptimalCheck (Appendix B). P & H results of a comparison of the author’s methods (random sample and local search) with Michalak’s methods (gradient and evolutionary algorithm) [18] are given in Appendix 8.

7.1 Random Sample

7.1.1 Algorithm

We will use the points from \( \{ p_t \}_{t=1}^{N_p} \) as the candidates for \( p \).

The idea of this sampling method is to fix each \( p_t \) as the point for \( \Sigma \) and then uniformly sample the unit sphere for a normal \( n \) of \( \Sigma \). We are essentially rotating the Poincaré plane about each point \( p_t \) from the flow or embedded map sequence. Figure 7.1 provides a flow map of the algorithm and below we give a more detailed outline.

Step 0: Initialization:
7.1. RANDOM SAMPLE

Figure 7.1: Flow Map of the Random Sample Method.
7.1. RANDOM SAMPLE

- We arbitrarily choose the number of points $N_{sp}$ to sample uniformly on the unit sphere.
- Initialize the Poincaré point index $k = 1$.

Step 1: Fix Point of Plane

- Fix $p_t$ as the point of the plane.

Step 2: Sample Unit Sphere for Normal:

We generate $N_{sp}$ sample points uniformly on the unit sphere (see Appendix C) as possible normals for $\Sigma$, namely:

$$S = \{ n_i : i = 1, 2, 3, \ldots, N_{sp}, n_i \in \mathbb{R}^3 \text{ and } |n_i| = 1 \}$$

For all $n_i \in S$, we compute the objective function $\Delta(p_t, n_i)$ given in Eq. 4.1.

Step 5: Update Optimal Poincaré Plane and Stopping Criteria:

- Update the normal $n$ and point $p$ of $\Sigma$ to the plane with the highest objective value:

$$\exists (p_t, n_i) \in \mathbb{R}^3 \times S, \Delta(p_t, n_i) \geq \Delta(p, n) \implies (p, n) = \arg \max_{(p_t, n_i)} \Delta(p_t, n_i)$$

Otherwise, there is no update.
- Update Poincaré point index $k = k + 1$ and return to Step 1.
- When the Poincaré point index $k > N_p$, we have exhausted all points of the flow or embedded map sequence and we stop.
7.2 Local Sphere Cap Search

7.2.1 Algorithm

We will use the points from \( \{p_t\}_{t=1}^{N_p} \) as the candidates for \( p \).

The idea of this local search method is to fix each \( p_t \) as the point for \( \Sigma \) and then search the unit sphere for a normal \( n \) of \( \Sigma \) using spherical caps. Recall from the analysis of the search space in Section, that we search within spherical caps because they resemble the well-defined regions with high values of the objective function \( \Delta \).

We are essentially rotating the Poincaré plane about each point \( p_t \) from the flow or embedded map sequence. Figure 7.2 provides a flow map of the algorithm and below we give a more detailed outline.
7.2. LOCAL SPHERE CAP SEARCH

Figure 7.2: Flow Map of the Local Sphere Cap Search Algorithm.
7.2. LOCAL SPHERE CAP SEARCH

Figure 7.3: Cap of unit sphere in \( \mathbb{R}^3 \), denoted by \( C_i \).

For the unit sphere in \( \mathbb{R}^3 \), let us denote the surface area of a cap of height \( h \), radius \( r_c \), angle \( \phi_c \), and center \( n_i \) by \( C_i \), as shown in Figure 7.3.

**Step 0: Initialization:**

- We arbitrarily choose the number of caps \( N_c \) and number of points \( N_{cp} \) to sample in each cap. We use a large, medium, and small size of spherical caps for our search, each defined by a different angle \( \phi_c \). By default, we chose \( \phi_c = \frac{\pi}{4} \), \( \phi_c = \frac{\pi}{8} \), and \( \phi_c = \frac{\pi}{16} \), respectively. The sequence of caps \( \{ C_i \}_{i=1}^{N_c} \) is partitioned into large, medium and small caps:

  \[
  \{ C_i \}_{i=1}^{N_c} = \left \{ C_1, \ldots, C_{\frac{N_c}{2}} \right \} \cup \left \{ C_{\frac{N_c}{2}+1}, \ldots, C_{\frac{3N_c}{4}} \right \} \cup \left \{ C_{\frac{3N_c}{4}+1}, \ldots, C_{N_c} \right \}
  \]

  \( \text{(7.1)} \)

The purpose in decreasing the size of our spherical caps is to attempt to converge to the global maximum of the unit sphere centered at \( p \), while still allowing to escape
7.2. LOCAL SPHERE CAP SEARCH

local maxima at the beginning of the optimization process for each fixed $p$.

- Initialize the Poincaré point index $k = 1$.

**Step 1: Fix Point of Plane and Initialize Normal:**

- Fix $p_t$ as the point of the plane.
- Initialize the cap index $i = 1$. We may choose $n_1$ in a number of different ways: uniformly on the unit sphere, PCAmin vector, or PCAmax vector. By default, we choose $n_1$ uniformly on the unit sphere.

**Step 2: Search Cap for Normal:**

We generate $N_{cp}$ sample points uniformly on the spherical cap $C_i$ (see Appendix D), namely:

$$S = \{n_{ij} \mid j = 1, 2, 3, \ldots, N_{cp} \text{ and } n_{ij} \in C_i\}$$

For all $n_{ij} \in S$, we compute the objective function $\Delta(p_t, n_{ij})$ given in Eq. 4.1.

**Step 3: Update Normal and Cap:**

- Update the centre of the cap $n_i$ to the sampled point $n_{ij}$ in the cap with the highest objective value:

$$\exists n_{ij} \in S, \Delta(p_t, n_{ij}) \geq \Delta(p_t, n_i) \implies n_i = \arg\max_{n_{ij}} \Delta(p_t, n_{ij})$$

Otherwise, there is no update.

- Update the size of the spherical cap according to the current iteration number and partition given in 7.1.

- Update cap index $i = i + 1$ and continue to search the unit sphere (Step 2) until $i = N_c$. 
7.2. LOCAL SPHERE CAP SEARCH

**Step 4: Restart Initialize Normal:**

As with the *Random Sample* method, we allow for a restart process, where $p_t$ remains fixed and we re-initialize the normal as in *Step 1*. The number of restarts is chosen arbitrarily, and by default we use 4 restarts.

**Step 5: Update Optimal Poincaré Plane and Stopping Criteria:**

- Update the normal $n$ and point $p$ of $\Sigma$ to the plane with the highest objective value:

  $$\exists (p_t, n_i) \in \mathbb{R}^3 \times S, \Delta(p_t, n_i) \geq \Delta(p, n) \implies (p, n) = \arg\max_{(p_t, n_i)} \Delta(p_t, n_i)$$

- Update *Poincaré point index* $k = k + 1$ and return to *Step 1*.

- When the *Poincaré point index* $k > N_p$, we have exhausted all points of the flow or embedded map sequence and we stop.
7.2. LOCAL SPHERE CAP SEARCH

7.2.2 Illustration of the method converging

In this example, we apply the Local Sphere Cap Search method to the Logistic Map with noise $\eta = 0.1$. We show steps 0-3 for the fixed point $p = (0.93, 0.23, 0.69)$, with no restart. Figures 7.4 - 7.6 illustrate the method converging, where black dots represent the centers of the chosen spherical caps, red dots are the points uniformly sampled on these caps, and all points are shown on the sphere gradient map centered at $p$.

In Figure 7.4, we first select the initial normal $n_1$ uniformly on the unit sphere. We then sample $N_{cp}$ points in the spherical cap centered at $n_1$. Next, the point $n_2$ is selected amongst these sample points as having the highest objective function value. We then sample $N_{cp}$ points in the spherical cap centered at $n_2$ and of these

Figure 7.4: **Frame 1**: Local Sphere Cap Search Algorithm converging for Logistic Map, with noise $\eta = 0.1$ and fixed point $p = (0.93, 0.23, 0.69)$. 
7.2. LOCAL SPHERE CAP SEARCH

$n_3$ is selected as having the highest objective function value. We continue this process and reach the cap centered at $n_4$.

Figure 7.5: **Frame 2:** Local Sphere Cap Search Algorithm converging for Logistic Map, with noise $\eta = 0.1$ and fixed point $p = (0.93, 0.23, 0.69)$.

Figures 7.4 and 7.5 show that the initial normal $n_1$ was chosen in a poor region for the objective function and that the algorithm’s search path has moved to the light blue region of higher objective function values by the 4$^{th}$ iteration of caps.

We continue the process of moving to new caps and sampling them. In Figure 7.6, we see that after the 7$^{th}$ iteration of caps, the algorithm is set to converge on the global maximum of the fixed unit sphere centered at $p = (0.93, 0.23, 0.69)$. 
Figure 7.6: **Frame 3:** Local Sphere Cap Search Algorithm converging for Logistic Map, with noise $\eta = 0.1$ and fixed point $p = (0.931466816, 0.229807252, 0.68616601)$. 
Chapter 8

Results of Experiments:
Comparing P&H optimization methods

The experimental results presented here are the outputs of P&H Method using the *intersect* and *slice* method for all twelve data sets from Chapter 3. The author’s methods (*random sample* and *local sphere cap search*) were compared with Michalak’s methods (*gradient* and *evolutionary algorithm*) [18].

The experiments are analogous to those completed by Michalak [18]. For each method, we optimize the Poincaré Plane 10 times and select the result with the maximum value of $\Delta(n,p)$. The data sets used are the same systems used in Michalak’s experiments [18]. For the stochastic time series, no noise was added. For the chaotic time series, maximum noise of $\eta = 0.1$ was added to the data.

To test Michalak’s methods, the program *Poincaré sections optimization tool (ps_opt)* was downloaded from his homepage [20]. Thus we used his exact implementation with default parameters for the *gradient* and *evolutionary* methods.
8.1 Chaotic Time Series Graph Results

Figure 8.1: Output of P&H Method using the “slice” method for the Chen-Lee time series; noise $\eta = 0.1$. The author’s methods (random sample and local search) were compared with Michalak’s methods (gradient and evolutionary algorithm).
8.1. CHAOTIC TIME SERIES GRAPH RESULTS

Figure 8.2: Output of P&H Method for the Bouali times series; noise $\eta = 0.1$. The author’s methods (random sample and local search) were compared with Michalak’s methods (gradient and evolutionary algorithm).

Figure 8.3: Output of P&H Method for the Logistic Map; noise $\eta = 0.1$. The author’s methods (random sample and local search) were compared with Michalak’s methods (gradient and evolutionary algorithm).
8.1. CHAOTIC TIME SERIES GRAPH RESULTS

Figure 8.4: Output of P&H Method for the Lorenz times series; noise $\eta = 0.1$. The author’s methods (random sample and local search) were compared with Michalak’s methods (gradient and evolutionary algorithm).

Figure 8.5: Output of P&H Method for the Rabinovich-Fabrikant times series; noise $\eta = 0.1$. The author’s methods (random sample and local search) were compared with Michalak’s methods (gradient and evolutionary algorithm).
Figure 8.6: Output of P&H Method for the Rössler times series; noise $\eta = 0.1$. The author’s methods (random sample and local search) were compared with Michalak’s methods (gradient and evolutionary algorithm).

8.2 Stochastic Time Series Graph Results
8.2. STOCHASTIC TIME SERIES GRAPH RESULTS

Figure 8.7: Output of P&H Method for the Guassian 1D times series. The author’s methods (random sample and local search) were compared with Michalak’s methods (gradient and evolutionary algorithm).

Figure 8.8: Output of P&H Method for the Guassian 3D times series. The author’s methods (random sample and local search) were compared with Michalak’s methods (gradient and evolutionary algorithm).
8.2. STOCHASTIC TIME SERIES GRAPH RESULTS

Figure 8.9: Output of P&H Method for the Uniform 1D times series. The author’s methods (**random sample** and **local search**) were compared with Michalak’s methods (**gradient** and **evolutionary algorithm**).

Figure 8.10: Output of P&H Method for the Uniform 3D times series. The author’s methods (**random sample** and **local search**) were compared with Michalak’s methods (**gradient** and **evolutionary algorithm**).
8.2. STOCHASTIC TIME SERIES GRAPH RESULTS

Figure 8.11: Output of P&H Method for the Random Walk 1D times series. The author’s methods (random sample and local search) were compared with Michalak’s methods (gradient and evolutionary algorithm).

Figure 8.12: Output of P&H Method for the Random Walk 3D times series. The author’s methods (random sample and local search) were compared with Michalak’s methods (gradient and evolutionary algorithm).
Chapter 9

Results of Experiments: P&H

Local Search Tests

The experimental results presented here are the outputs of the P&H Method using the author’s method local search (intersection) with the objective function $\Delta(n, p)$, for all twelve data sets from Chapter 3. For the chaotic data sets, the tests were performed with no noise and with noise $\eta = 0.1$. 30 tests of this method were completed for each individual time series and used to calculate the mean and standard deviation. In 9.1 the tests use the objective function $\Delta(n, p)$ in optimizing the Poincaré Plane, and in 9.2 the tests use the objective function $\psi(n, p)$. 
9.1. 30 TESTS WITH $\Delta(N, P)$

<table>
<thead>
<tr>
<th>Stochastic Times Series (noise = 0; intersect method)</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Map/Flow</td>
<td>Mean</td>
<td>St. deviation</td>
</tr>
<tr>
<td>Gaussian 1D</td>
<td>22.21</td>
<td>0.10</td>
</tr>
<tr>
<td>Gaussian 3D</td>
<td>22.69</td>
<td>0.03</td>
</tr>
<tr>
<td>Uniform 1D</td>
<td>33.88</td>
<td>0.05</td>
</tr>
<tr>
<td>Uniform 3D</td>
<td>36.86</td>
<td>0.08</td>
</tr>
<tr>
<td>Random Walk 1D</td>
<td>25.79</td>
<td>0.03</td>
</tr>
<tr>
<td>Random Walk 3D</td>
<td>0.17</td>
<td>0.00</td>
</tr>
<tr>
<td>Mean</td>
<td>23.60</td>
<td>0.05</td>
</tr>
<tr>
<td>Min./Max.</td>
<td>0.17/36.86</td>
<td></td>
</tr>
</tbody>
</table>

Figure 9.1: 30 tests of the P&H Method on all 6 stochastic data sets using the local search method, the objective function $\Delta(n, p)$ (Eq. (4.1)), and the “intersect” method. Results of P&H Method were then used to calculate the mean and standard deviation.

9.1 30 tests with $\Delta(n, p)$
### 9.1. 30 TESTS WITH $\Delta(N, P)$

#### Figure 9.2: 30 tests of the P&H Method on all 6 chaotic data sets (noise $\eta = 0.1$) using the local search method, the objective function $\Delta(n, p)$ (Eq. 4.1), and the “intersect” method. Results of P&H Method were then used to calculate the mean and standard deviation.

<table>
<thead>
<tr>
<th>Map/Flow</th>
<th>Mean</th>
<th>St. deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chen-Lee</td>
<td>222.96</td>
<td>0.81</td>
</tr>
<tr>
<td>Bouali</td>
<td>65.99</td>
<td>0.85</td>
</tr>
<tr>
<td>Logistic</td>
<td>62.50</td>
<td>1.02</td>
</tr>
<tr>
<td>Lorenz</td>
<td>77.62</td>
<td>0.28</td>
</tr>
<tr>
<td>Rabinovich-Fabrikant</td>
<td>139.36</td>
<td>1.99</td>
</tr>
<tr>
<td>Rossler</td>
<td>255.69</td>
<td>3.51</td>
</tr>
<tr>
<td><strong>Mean</strong></td>
<td>137.35</td>
<td>1.41</td>
</tr>
<tr>
<td><strong>Min./Max.</strong></td>
<td>62.5/255.69</td>
<td></td>
</tr>
</tbody>
</table>

#### Figure 9.3: 30 tests of the P&H Method on all 6 chaotic data sets (no noise) using the local search method, the objective function $\Delta(n, p)$ (Eq. 4.1), and the “intersect” method. Results of P&H Method were then used to calculate the mean and standard deviation.

<table>
<thead>
<tr>
<th>Map/Flow</th>
<th>Mean</th>
<th>St. deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chen-Lee</td>
<td>667.20</td>
<td>0.06</td>
</tr>
<tr>
<td>Bouali</td>
<td>309.96</td>
<td>0.19</td>
</tr>
<tr>
<td>Logistic</td>
<td>235.18</td>
<td>0.06</td>
</tr>
<tr>
<td>Lorenz</td>
<td>500.85</td>
<td>13.70</td>
</tr>
<tr>
<td>Rabinovich-Fabrikant</td>
<td>456.51</td>
<td>16.84</td>
</tr>
<tr>
<td>Rossler</td>
<td>1185.65</td>
<td>0.85</td>
</tr>
<tr>
<td><strong>Mean</strong></td>
<td>559.24</td>
<td>5.28</td>
</tr>
<tr>
<td><strong>Min./Max.</strong></td>
<td>235.18/1185.65</td>
<td></td>
</tr>
</tbody>
</table>
9.2 30 tests with $\psi(N, P)$

<table>
<thead>
<tr>
<th>Stochastic Times Series (noise = 0; intersect method)</th>
<th>Mean</th>
<th>St. deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gaussian 1D</td>
<td>6.05</td>
<td>0.0080</td>
</tr>
<tr>
<td>Gaussian 3D</td>
<td>6.13</td>
<td>0.0133</td>
</tr>
<tr>
<td>Uniform 1D</td>
<td>6.65</td>
<td>0.0216</td>
</tr>
<tr>
<td>Uniform 3D</td>
<td>7.92</td>
<td>0.0082</td>
</tr>
<tr>
<td>Random Walk 1D</td>
<td>6.70</td>
<td>0.0040</td>
</tr>
<tr>
<td>Random Walk 3D</td>
<td>1.62</td>
<td>0.0037</td>
</tr>
<tr>
<td>Mean</td>
<td>5.84</td>
<td>0.0098</td>
</tr>
<tr>
<td>Min./Max.</td>
<td>1.62/7.92</td>
<td></td>
</tr>
</tbody>
</table>

Figure 9.4: 30 tests of the P&H Method on all 6 stochastic data sets using the local search method, the objective function $\psi(n, p)$ (Eq. 4.4), and the “intersect” method. Results of P&H Method were then used to calculate the mean and standard deviation.
9.2. 30 TESTS WITH $\psi(N, P)$

<table>
<thead>
<tr>
<th>Chaotic Times Series (noise = 0.1; intersect method)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Map/Flow</strong></td>
</tr>
<tr>
<td>Chen-Lee</td>
</tr>
<tr>
<td>Bouali</td>
</tr>
<tr>
<td>Logistic</td>
</tr>
<tr>
<td>Lorenz</td>
</tr>
<tr>
<td>Rabinovich-Fabrikant</td>
</tr>
<tr>
<td>Rossler</td>
</tr>
<tr>
<td><strong>Mean</strong></td>
</tr>
<tr>
<td><strong>Min./Max.</strong></td>
</tr>
</tbody>
</table>

Figure 9.5: 30 tests of the P&H Method on all 6 chaotic data sets (noise $\eta = 0.1$) using the local search method, the objective function $\psi(n, p)$ (Eq. 4.4), and the “intersect” method. Results of P&H Method were then used to calculate the mean and standard deviation.

<table>
<thead>
<tr>
<th>Chaotic Times Series (noise = 0; intersect method)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Map/Flow</strong></td>
</tr>
<tr>
<td>Chen-Lee</td>
</tr>
<tr>
<td>Bouali</td>
</tr>
<tr>
<td>Logistic</td>
</tr>
<tr>
<td>Lorenz</td>
</tr>
<tr>
<td>Rabinovich-Fabrikant</td>
</tr>
<tr>
<td>Rossler</td>
</tr>
<tr>
<td><strong>Mean</strong></td>
</tr>
<tr>
<td><strong>Min./Max.</strong></td>
</tr>
</tbody>
</table>

Figure 9.6: 30 tests of the P&H Method on all 6 chaotic data sets (no noise) using the local search method, the objective function $\psi(n, p)$ (Eq. 4.4), and the “intersect” method. Results of P&H Method were then used to calculate the mean and standard deviation.
Chapter 10

New P&H Criterion

The P&H Method currently distinguishes between chaotic and stochastic times series by the qualitative pattern produced in the P&H graph. Recall from Section 2.3 that chaotic time series produce increasing and decreasing jagged peaks, whereas stochastic time series produce a smooth, decreasing curve. We have found that ideally this is the case for some time series, but for others this qualitative criterion fails to distinguish between chaotic and stochastic time series. See Chapter 8 for results of the author’s methods (random sample and local sphere cap search), compared with Michalak’s methods (gradient and random sample), applied to the twelve data sets from Chapter 3.

The claim in this thesis is that instead of using the qualitative criterion of the P&H graph to determine if a time series was generated by a deterministic chaotic process or by a stochastic one, a single quantitative criterion can be obtained from the graph to do so. We propose that the quantitative criterion of the P&H graph is the objective function $\Delta$ from Chapter 4. We explain the new criterion in detail below.
10.1 P&H Quantitative Criterion

For a flow or embedded map $\Gamma = \{p_i\}_{i=1}^{N_p} \in \mathbb{R}^3$, the following criterion distinguishes between deterministic and stochastic dynamical systems:

1. Apply the P&H Method to the flow or embedded map $\Gamma \in \mathbb{R}^3$. When optimizing the Poincaré Plane $\Sigma$, the objective function $\Delta$ from Eq. (4.1) is used.

2. The final value $\Delta_\Sigma$ obtained from the optimized Poincaré Plane $\Sigma$ is attributed to the flow/embedded map $\Gamma$.

3. The criterion distinguishes the flow/embedded map $\Gamma$ (and, if applicable, the original map) to be:

$$
\begin{cases} 
\text{stochastic (unpredictable)} & \text{if } \Delta_\Sigma(n, p) \leq \hat{\Delta} \\
\text{deterministic (predictable)} & \text{if } \Delta_\Sigma(n, p) > \hat{\Delta}
\end{cases}
$$

where $\hat{\Delta}$ is a fixed value, determined by values of tested time series that are known to be chaotic or stochastic. Currently, we propose that $\hat{\Delta} = 50$, being the midpoint between the minimum value of $\Delta$ obtained from tested chaotic data sets and maximum value of $\Delta$ obtained from tested stochastic data sets.
10.2 Justification for New Criterion

In Chapter 9, we presented the results of the outputs of the P&H Method using the author’s method *local search (intersection)* with the objective function $\Delta(n,p)$, for all twelve data sets from Chapter 3. The purpose of these tests is to test the precision of the new methods and to establish a quantitative difference in the results of the P&H Method.

Since we completed 30 tests, the Central Limit Theorem [13] allows us to construct a normal distribution for the results of each individual time series sample of size $n = 30$. Figure 10.1 provides the collection of distributions found from the tests for the stochastic time series and the chaotic time series with noise $\eta = 0$. The full results are given in Chapter 9.

We can see clear separation between the stochastic time series and the chaotic time series with no noise, but we even have excellent separation when there is noise of $\eta = 0.1$, as shown in Figure 10.1. $\hat{\Delta} = 50$ used in the P&H Quantitative Criterion is shown between the two sets. $\hat{\Delta}$ is the median of the minimum sample distribution mean for tested chaotic data sets with noise $\eta = 0.1$ and the maximum sample distribution mean for tested stochastic data sets.

$$\hat{\Delta} = \frac{36.86 + 62.50}{2} \approx 50$$  \hspace{1cm} (10.1)
10.2. JUSTIFICATION FOR NEW CRITERION

Figure 10.1: Summary of Results from Chapter [9]
Normal distribution for each individual time series using the local search (intersection) method and objective function $\Delta(n, p)$. $\hat{\Delta} = 50$ used in the P&H Quantitative Criterion to distinguish between chaotic and stochastic time series is shown between the two sets of distributions.
Chapter 11

Conclusions and Future Research

11.1 Conclusion

We can conclude from our results in Chapter 8 that the new optimization methods (random sample and local search) for optimization of the Poincaré Plane outperform Michalak’s methods (gradient and evolutionary algorithm). However, just as Michalak [18] discovered, better performing methods (random sample and local search) took longer to compute.

Of the two new methods, local search performed better than random sample for both the slice and intersect forms of the P&H Method. Interestingly, for most dynamical flows like the Chen-Lee system, the intersect method seemed to capture the dynamics of the systems/map than the slice method.

Some peculiarities are evident in the results of this thesis when adding maximum noise of $\eta = 0.1$ to the chaotic data sets. First, similar to Michalak’s observations [18], it can be seen in the P&H graphs that the zig-zag pattern is hard to detect in the Lorenz data set. Second, the slice method is much more effective than the intersect method in producing zig-zag patterns for the Logistic Map. Third, large
peaks, followed by logarithmic sections are observed in the P&H graphs of the Bouali data set.

In testing other objective functions, none were found to produce more distinctive zig-zag patterns for deterministic dynamical systems than \( \Delta \) (Eq. 4.1), that was proposed by Michalak [18]. \( \psi \) (Eq. 4.4) captured nearly the same distinctive zig-zag patterns, but \( \Delta \) still gave better separation between chaotic and stochastic time series for the \( P&H \) Quantitative Criterion.

The \( P&H \) Quantitative Criterion successfully distinguished between chaotic and stochastic time series and provides a quantitative value or property that can be attributed to a time series, based on the P&H method.

11.2 Future Research

There are many interesting questions which arise from this research and offer directions for future work. We mention a few of them below:

1. The entire motivation for this research project on the P&H Method is to further the GenPred [7] time series prediction method. The program OptimalCheck (Appendix B) provides a necessary analysis tool to optimize the Poincaré Plane. The GenPred method has been implemented in OptimalCheck and successful time series prediction should be the focus of future research related to this project.

2. The \( P&H \) Quantitative Criterion presented in this paper is based off the data sets that have been tested by the author and Krzysztof Michalak [18]. To ensure that this criterion can distinguish between any stochastic and chaotic time series, more tests should be done like those in this paper. The \( P&H \) Quantitative
11.2. FUTURE RESEARCH

Criterion should also be tested on any time series arising from natural phenomena.

3. We contend that the value of $\Delta_{\Sigma}$ from the P&H Quantitative Criterion presented in this paper is a property that can be attributed to a chaotic map. Most properties of a chaotic map, like its Lyapunov exponent, are invariant under a conjugate map (see Chapter [1]). The question then arises whether $\Delta_{\Sigma}$ is also. In particular, since the Logistic Map is conjugate to the Tent Map, one could apply the P&H Quantitative Criterion to the Tent Map and compare its value of $\Delta_{\Sigma}$ to that of the Logistic Map. If these values are equal, this would support the claim that $\Delta_{\Sigma}$ from the P&H Quantitative Criterion attributed to a chaotic map is invariant under a conjugate map.

4. From a theoretical standpoint, a greater understanding of the reasons why the P&H Method generates the distinctive patterns it does for chaotic and stochastic time series is much needed. Golestani et al., who originally propose the method, discuss this briefly [9]. They simply state that “the L(k) vector indicates a zigzag pattern for chaotic time series, because of the stretching and folding property.” This may seem intuitive, but requires a more direct connection between the stretching and folding of a map or flow and the pattern that arises in the P&H graph. This also brings to question as to why the method uses a modified form of the Higuchi dimension and not another measure of the dimension.

5. Golestani et al. [9] give no justification for the modification of the normalization factor in the Higuchi method from the original (Eq. 2.15) [12]:

$$\frac{N - m}{\left\lfloor \frac{N - m}{k} \right\rfloor} \cdot k$$
to that used in the P&H Method (Eq. 2.14):

\[
\frac{N - m}{\left\lfloor \frac{N - m}{k} \right\rfloor}
\]

Julan Al-Yassin [1], a fellow research partner, has indeed found that there seems to be a discrepancy as to which normalization factor to use. There are many instances in scientific papers where the modified normalization factor (Eq. 2.14) has been used to calculate the Higuchi fractal dimension, without any justification. This is a further research question to be answered.

6. Other methods of optimizing the Poincaré Plane could be implemented for the P&H Method, as well as the use of preprocessing methods such as noise removal techniques.
Appendix A

Runge Kutta Numerical Method

The *ode45* solver in MATLAB is an implementation of the Runge Kutta method for estimating solutions to ordinary differential equations. This method, in general, seeks to estimate subsequent points $y_{n+1}$ of a trajectory \[14\].

Given:

\[ y' = f(t, y) \]
\[ y_n = y(t_n) \]  \hspace{1cm} (A.1)
\[ y_0 = y(t_0) \]

we compute:

\[ f_1 = hf(t_n, y_n) \]
\[ f_2 = hf \left( t_n + \frac{h}{2}, y_n + \frac{k_1}{2} \right) \]
\[ f_3 = hf \left( t_n + \frac{h}{2}, y_n + \frac{k_2}{2} \right) \]  \hspace{1cm} (A.2)
\[ f_3 = hf \left( t_n + h, y_n + k_3 \right) \]
\[ y_{n+1} = y_n + \frac{1}{6} (k_1 + 2k_2 + 2k_3 + k_4) \]

where $h$ is a chosen step size.
Appendix B

C++ Program: OptimalCheck

B.1 Generating Random Numbers

At first glance it may seem a trivial task to produce random numbers through a machine or programming language. However, to produce truly random numbers is actually impossible without some outside input from the physical universe that has some form of entropy. Computers use random number generators, which are algorithms that, based on an initial seed or by means of continuous input, produces a sequence of numbers. It is important to make a distinction between pseudorandom, truly random, and real random numbers.

By real random numbers, we mean the independent realizations of a uniformly distributed random variable [28].

By truly random numbers, we mean output of a physical experiment which is considered to be random (entropic), like radioactive decay or atmospheric noise [28].

By pseudorandom number, we mean output of a random number generator that is indistinguishable from truly random numbers by any efficient algorithm [28].

For our methods of sampling for optimization or in generating stochastic time
series, it is sufficient to use pseudorandom numbers. Some pseudorandom number
generators in C++ and Excel, however, do not pass most statistical tests. We use the
Mersenne Twister (MT) that was developed by Matsumoto and Nishimura \[16\].
It was adopted into the Boost Library before being added into in the C Standard
Library and has passed several stringent statistical tests \[16\].

Our implementation of generating pseudorandom numbers in C++ is given in
Listing B.1. Here, we seed the MT with the registered time of compilation of the
program, and then pass the pseudorandom sequence of numbers to the built-in prob-
ability distributions of the C Standard Library.

Listing B.1: C++ Code generating pseudorandom points with Mersenne Twister and C-Standard
Normal Distribution that are uniformly distributed on the unit sphere in $\mathbb{R}^3$. \[25\]
B.2 Input File and Parameters

To use the optimization methods and analysis in the C++ program `optimalcheck`, the program will eventually be made available and users will be able to use it for further research. In Listing B.2, we give an example of an .txt input file, where the user is able to change the parameters of each optimization and analysis process. We give a brief summary of the parameters below.

Listing B.2: Input .txt file for `optimalcheck`

```plaintext
# Help Menu: ./optimalcheck.exe --help
# Example run: ./optimalcheck.exe --run-input-file testinputfile.Logistic1500(06.28.2017).txt

# =============== Main Parameters ===============
series-csv-filename = Logistic1500.csv
series-dimension = 1
add-noise-operation = false
noise-level = 0.1
operation = optimize_then_slide_window
optimization-method = PCAmin
k = 20
k1 = 20
lag = 1
window-size = 1498

# =============== Opt. Parameters ===============
optimize-iterations = 1498
```
Main Parameters:

**series-csv-filename** - name of .csv file that contains time series in first column. (std::string)

**series-dimension** - dimension of the state space of the time series. (std::int)

**operation** - the operation that *optimalcheck* will perform. Inputs include: “slidewindow”, “optimize”, “optimize_then_slidewindow”, “spheregradient”, etc. (std::string)

**optimization-method** - the optimization method that *optimalcheck* will perform. Inputs include: “random_search”, “random_sample”, “PCA_min”, “PCA_max”. (std::string)

**k** - the $k - max$ used in defining the set $\mathcal{K}$ used in the calculation of the Higuchi dimension. (std::int)

**k1** - the $k - max$ used in defining the set $\mathcal{K}$ used in the calculation of the second Higuchi dimension on the P&H graph. (std::int)
B.2. INPUT FILE AND PARAMETERS

**lag** - user defined lag \( \tau \) for embedding a 1-dimensional timeseries. (std::int)

**window-size** - The user defined number of points to analyze in a given time series. (std::int)

**Secondary Parameters:**

**num-gradient-points** - gradient maps: user defined local search/random sample: number points. (std::int)

**num-predict** - GenPred: number of user defined points to predict. (std::int)

**num-window-slides** - sliding window: number of user defined slides. (std::int)

**window-translation** - sliding window: user defined size of translation. (std::int)

**Optimization Parameters:**

**optimize-iterations** - local search/random sample: user defined number points. (std::int)

**num-random-sample** - random sample: user defined number points sampled on sphere. (std::int)

**num-caps** - local search: user defined number caps on sphere. (std::int)

**num-points-per-cap** - local search: user defined number points per cap. (std::int)

**num-restarts** - local search: user defined number restarts on sphere. (std::int)
Appendix C

Generating uniform points on the unit sphere

One way to generate uniform points on the unit sphere is to generate three normally distributed, independent, random variables:

\[
x \sim \mathcal{N}(0,1) \\
y \sim \mathcal{N}(0,1) \\
z \sim \mathcal{N}(0,1)
\] (C.1)

We then set these as the coordinates of a point in \(\mathbb{R}^3\) and normalize:

\[
(x', y', z') = \frac{(x, y, z)}{\sqrt{x^2 + y^2 + z^2}}
\] (C.2)

Jan Poland [24] shows that the points \((x', y', z')\) generated with this method will give a uniform distribution of points on the unit sphere.
Appendix D

Generating uniform points on a unit sphere cap

We can describe the sphere cap $C$ in Figure 7.3 with any one of the parameters $\phi_c$, $r_c$ or $h$ since the sphere has unit radius. We will use them interchangeably where convenient. We will use polar coordinates to describe points on the sphere. It would seem intuitive that we could generate polar coordinates $(\theta, \phi)$ uniform on $[0, 2\pi) \times [0, \phi_c)$ as in Figure D.1 and this would be uniform on the cap $C$. However, we can see in Figure D.2 that the points are more dense near the centre of the cap.

We must find a probability distribution function (p.d.f.) $f$ that is uniform on $C$. If so, then over time the probability of $f$ landing in any differential surface element $dA \subset C$ is one. The differential surface element is shown in Figure D.3.

Let $v \in C$. We want the p.d.f. $f$ to be constant for a uniform distribution and for the integration of $f$ over the surface $C$ will be one:

$$\int \int_C f(v) dA = 1$$
Figure D.1: Uniform distribution on $[0, 2\pi) \times [0, \phi_c)$. Source: [29]

Figure D.2: Uniform distribution on $[0, 2\pi) \times [0, \phi_c)$, together with $\rho = 1$ are mapped to polar coordinates on $C$. The points are denser near the top of the sphere cap and hence the distribution is not uniform on $C$. 
Source: [29]
Figure D.3: $dA$ of differential surface element in spherical coordinates. Source: [29]

$$\Rightarrow \int \int_{C} dA = \frac{1}{f(v)}$$

Since the surface area of $C$ is given by $\int \int_{C} dA = \pi (h^2 + r_c^2)$, we have:

$$\frac{1}{f(v)} = \pi (h^2 + r_c^2)$$

$$\Rightarrow f(v) = \frac{1}{\pi (h^2 + r_c^2)}$$

We wish to parametrize $f$ as $f(\theta, \phi)$ where:

$$f(v) dA = \frac{1}{\pi (h^2 + r_c^2)} dA = f(\theta, \phi) d\theta d\phi$$

but for polar coordinates, we have $dA = r^2 \sin \phi d\theta d\phi$ and since we are in the unit
circle \( r = 1 \), thus:

\[
dA = \sin \phi \, d\theta \, d\phi \quad \text{(D.1)}
\]

\[
\Rightarrow f(\theta, \phi) \, d\theta \, d\phi = \frac{1}{\pi (h^2 + r_c^2)} \cdot (\sin \phi \, d\theta \, d\phi) \quad \text{(D.2)}
\]

\[
\Rightarrow f(\theta, \phi) = \frac{\sin \phi}{\pi (h^2 + r_c^2)} \quad \text{(D.3)}
\]

We can marginalize the joint p.d.f. given in Eq. 4.1 as follows:

\[
f(\phi) = \int_0^{2\pi} f(\theta, \phi) \, d\theta
\]

\[
= \int_0^{2\pi} \frac{\sin \phi}{\pi (h^2 + r_c^2)} \, d\theta
\]

\[
= \frac{(2\pi) \sin \phi}{\pi (h^2 + r_c^2)}
\]

\[
= \frac{2 \sin \phi}{h^2 + r_c^2} \quad \text{(D.4)}
\]

\[
f(\theta) = \int_0^{\phi_c} f(\theta, \phi) \, d\phi
\]

\[
= \int_0^{\phi_c} \frac{\sin \phi}{\pi (h^2 + r_c^2)} \, d\phi
\]

\[
= \frac{1}{\pi (h^2 + r_c^2)} \cdot (-\cos \phi) \bigg|_{0}^{\phi_c}
\]

\[
= \frac{1 - \cos \phi_c}{\pi (h^2 + r_c^2)}
\]

Now we see the reason why the uniform distribution on \([0, 2\pi] \times [0, \phi_c]\) does not generate uniform points on \( C \). The marginal distribution \( f(\theta) \) is constant and hence uniform, however, the marginal distribution \( f(\phi) \) still depends on \( \sin \phi \). We can use the Inverse Transform Sampling Method \[29\] to sample numbers \( \phi \) that will result in the distribution given by \( f(\phi) = \frac{2 \sin \phi}{h^2 + r_c^2} \). For this method, we need the
cumulative distribution function (c.d.f.) with respect to $\phi$:

$$F(\phi) = \int_{0}^{\phi} f(\phi) d\phi$$

$$= \int_{0}^{\phi} \frac{2 \sin \phi}{h^2 + r_c^2} d\phi$$

$$= \left[ \frac{2}{h^2 + r_c^2} \cdot (-\cos \phi) \right]_{0}^{\phi}$$

$$= \frac{2(1 - \cos \phi)}{h^2 + r_c^2}$$

(D.6)

Because the c.d.f. $F(\phi)$ is a monotonically strictly increasing function from $[0, \phi_c] \rightarrow [0, 1]$, it is invertible. Its inverse is given by:

$$F^{-1}(U) = \cos^{-1} \left[ 1 - U \left( \frac{h^2 + r_c^2}{2} \right) \right]$$

(D.7)

Using some simple trigonometry from Figure D.4, we can replace $h$ and $r_c$ with $\phi_c$:

$$\cos \phi_c = 1 - h$$

$$h = 1 - \cos \phi_c$$

(D.8)
Pythagorean Theorem gives:

\[(1 - h)^2 + a^2 = 1\]

\[a^2 = 1 - (1 - h)^2\]

\[a^2 = 2h - h^2\]

\[a = \sqrt{2h - h^2}\]  \hspace{1cm} (D.9)

Equations \text{[D.8]} and \text{[D.9]} simplify our c.d.f. inverse:

\[F^{-1}(U) = \cos^{-1} \left[ 1 - U \left( \frac{h^2 + r_c^2}{2} \right) \right]
\]

\[= \cos^{-1} \left[ 1 - U \left( \frac{h^2 + (\sqrt{2h - h^2})^2}{2} \right) \right]
\]

\[= \cos^{-1} \left[ 1 - U \left( \frac{h^2 + 2h - h^2}{2} \right) \right] \hspace{1cm} (D.10)
\]

\[= \cos^{-1} [1 - U(h)]
\]

\[= \cos^{-1} [1 - U (1 - \cos \phi_c)]
\]

Cory Simon \cite{Simon2018} explains how Inverse Transform Sampling works in this case. Let \(U(0,1)\) be a uniform number on \((0,1)\). Recall that \(F(\phi)\) is the c.d.f. of the marginal p.d.f. \(f(\phi)\). We then have the following probability:

\[Pr (U \leq F(\phi)) = F(\phi) \hspace{1cm} (D.11)\]

Because \(F(\phi)\) is strictly positive and using its inverse given in Equation \text{[D.10]}, the inequality is preserved and we have:

\[Pr (F^{-1}(U) \leq \phi) = F(\phi) \hspace{1cm} (D.12)\]
This shows that $F(\phi)$ is the c.d.f. of the random variable $F^{-1}(U)$ and hence follows the same distribution as $\phi$. Thus, the algorithm for generating a uniform point, in polar coordinates $(\theta, \phi, \rho)$ on the unit sphere cap $C$ is as follows:

- Generate a uniform random number $u$ from the distribution $U(0, 1)$
- Compute $\phi$ such that $F(\phi) = u$, i.e. using inverse c.d.f. found in Eq. [D.10]:
  $$F^{-1}(u) = \cos^{-1} \left[ 1 - u (1 - \cos \phi_c) \right]$$
- This $\phi$ is a random number drawn from the marginal distribution $f(\phi)$ found in Eq. [D.4]
  $$f(\phi) = \frac{2 \sin \phi}{h^2 + r_c^2}$$
- Finally, we set $\rho = 1$ and generate $\theta \sim U(0, 2\pi)$, independently

We can see in Figure [D.5] that the points generated from $f(\phi)$ are more dense around $\phi = \phi_c$ to account for the density near the top of the cap when $\phi = 0$. This leads to the uniform distribution of points on the cap $C$ shown in Figure [D.6]
Figure D.6: Inverse Transform Sampling leads to uniformly distributed points on sphere cap $C$. Source: [29]
Bibliography


Vita Auctoris

Mr. Jeremy George Cavers was born in 1990, in Windsor, Canada. He completed the Concurrent Mathematics program at the University of Windsor in 2013. He earned his Honours Bachelor of Science in Mathematics and Honours Bachelor of Arts in Education, with a minor in History. He expects to graduate with a Master of Science degree in Mathematics in Summer 2017 and is to pursue an M.D. at University College Cork in Fall 2017.