



university of
 groningen

faculty of science
and engineering

mathematics and applied
mathematics

A toy model for the El Niño-Southern Oscillation

Master Project Mathematics SBP-track

June 2020

Student: G. Pomstra

First supervisor: dr. A.E. Sterk

Second assessor: dr. ir. F.W. Wubs

Abstract

The ENSO model by Timmerman and Jin (2002) models the El Niño-Southern Oscillation using 3 prognostic differential equations. In this thesis, we use one of the trajectories of this ENSO model and consider a Poincaré section given by the local maxima of eastern Pacific sea surface temperatures. The relation of the resulting sequence of values can be approximated with 1 dimensional maps, with which we will examine bifurcations, snap-back repellers and predictability of extreme events. Lastly, we conclude that employing a toy model can be useful since it reduces, for example, time and calculation capacity while we still obtain similar results for the chosen variable.

Contents

1	Introduction	5
2	Preparatory dynamical systems theory	8
2.1	General terminology	8
2.2	Basic concepts on mappings	9
2.3	Time series and Poincaré maps	9
2.4	Fixed points, periodic points and their stability	10
2.5	Snap-back repellers	14
2.6	Bifurcation theory	15
3	The toy model and its map approximations	17
3.1	Derivation of the toy model	17
3.2	A piece-wise linear approximation	19
3.3	A piece-wise linear approximation with a shifting point	20
3.4	A piece-wise third degree polynomial approximation	21
4	Basic dynamics of the toy model	22
4.1	General properties of the approximations	22
4.2	Basic properties of linear approximations	25
4.3	Basic properties of the polynomial approximation	28
5	Bifurcation of the linear approximation with shifting top point	30
5.1	Period tripling	30
5.2	Chaotic behaviour inside branches	30
5.3	Period 10 solutions and the wall of chaos	31
6	Snap-back repellers	36
6.1	Finding snap-back repellers for the linear approximation	36
6.2	Proving chaos in the sense of Devaney with the snap-back repellers	37
6.3	Snap-back repeller in the linear approximations with shifting point	37
7	Predictability of extreme events	39
7.1	Applying the methodology on the toy model	39
7.2	Insight in predictability calculations	40
7.3	Predictability of extreme events of linear approximations	41
8	Discussion and conclusion	44
	References	46
	Appendix	47
	Proof of Corollary 1	47
	Matlab code	49
	Index	79

Acknowledgements

I would like to thank my supervisor dr. A.E. Sterk for his help and support with my research.

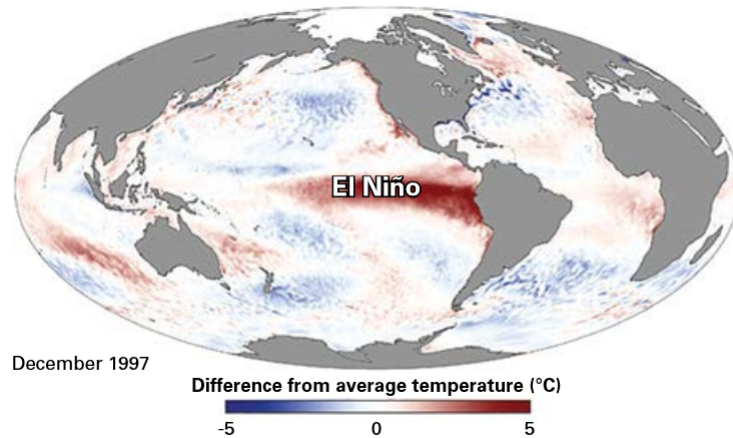


Figure 1: An example of El Niño in December 1997. A tongue of warm water is visible in red at the left of South-America [WMO14].

1 Introduction

In this thesis, we will derive and analyse a toy model for the El Niño-Southern Oscillation. The models that will be discussed analyze the El Niño phenomenon using dynamical systems theory.

The toy model we will construct will be based on data obtained from a model by Timmerman and Jin. This is a continuous dynamical system governed by 3 prognostic differential equations that we will refer to as the ENSO model throughout the thesis. The toy model we will create will be a 1 dimensional discrete dynamical system.

Before we focus on the toy model, we will first give a short introduction to the El Niño phenomenon and the ENSO model.

A short introduction to El Niño

El Niño is a weather phenomenon occurring in the central Pacific ocean and it is due to a circulation of sea surface temperatures called the El Niño-Southern Oscillation. This oscillation takes 2 to 7 years to circulate. El Niño got its name by Peruvian fishermen who named it “boy child” because the high temperatures in that region appear just before Christmas.

An example shown in Figure 1 is the El Niño of 1997 to 1998. In the red area, sea surface temperatures rose to 5 degrees above average. This powerful case of El Niño led to extreme rainfall in Kenya, Somalia and California, severe drought in Indonesia and lead to the warmest year in recorded history up until then.

El Niño events start with a warming of the surface water by the sun near the equator, which causes more clouds and rain as shown in Figure 2. Normally, trade winds blow warm water to the west, like with La Niña. With El Niño, these trade winds weaken and even reverse on the west side of the ocean. Warm water stays on the surface of the east side and is not replaced by cool water from a cold upwelling. Therefore the thermocline, the transition layer of water with different temperatures and density, on the east side is deeper than in usual circumstances.

The outlines of the ENSO model

The model we referred to is the low-order model for the El Niño-Southern Oscillation discussed by A. Timmermann and F.-F. Jin [TJ02, p.3-1]. For our studies we do not require to understand this model and therefore we only will briefly state the equations of the model. In fact, we will not use this model in its entire, only its results and occasionally we refer to its equations.

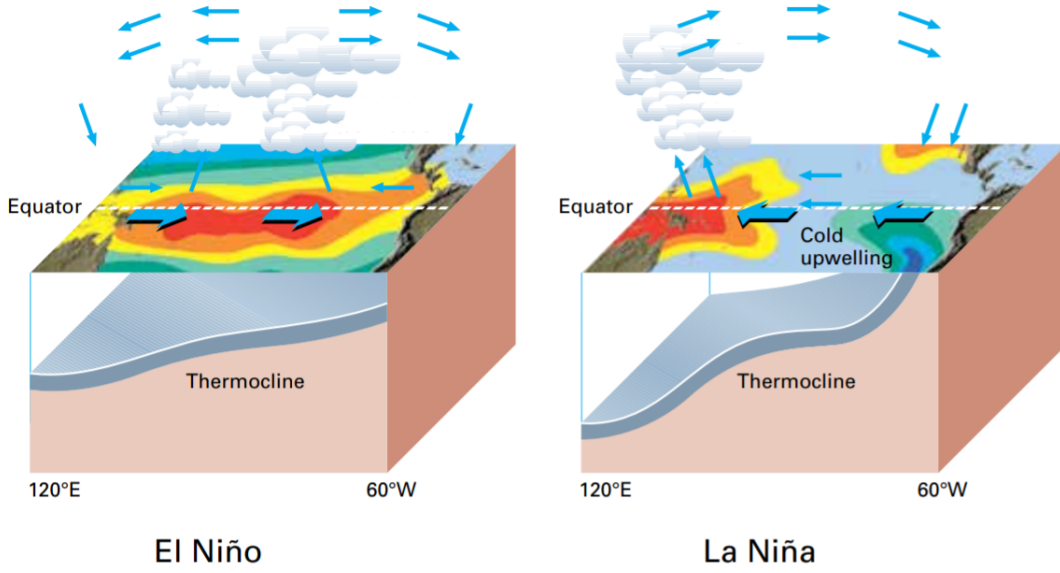


Figure 2: The physical process of extreme cases in the El Niño-Southern Oscillation [WMO14].

The **ENSO model** is a continuous dynamical system that consists of three prognostic equations which determine the rate of change of the western T_1 and eastern equatorial Pacific sea surface temperatures T_2 and the anomalous western equatorial thermocline depth h_1 .

$$\begin{aligned}\frac{dT_1}{dt} &= -\alpha(T_1 - T_r) - \frac{2u(T_2 - T_1)}{L}, \\ \frac{dT_2}{dt} &= -\alpha(T_2 - T_r) - \frac{w(T_2 - T_{sub})}{H_m}, \\ \frac{dh_1}{dt} &= r \left(-h_1 - \frac{bL\tau}{2} \right).\end{aligned}\tag{1}$$

Wind stress anomalies τ , zonal advection u and equatorial upwelling w in the eastern equatorial Pacific, the eastern equatorial thermocline depth h_2 are governed and subsurface temperature T_{sub} by equations (2).

$$\begin{aligned}u &= \frac{\varepsilon L \beta \tau}{2}, & w &= -H_m \beta \tau, \\ \tau &= \mu(T_1 - T_2) \frac{\sigma \zeta_t - 1}{\beta}, & h_2 &= h_1 + bL\tau, \\ T_{sub} &= T_r - \frac{T_r - T_{r0}}{2} \left(1 - \frac{\tanh(H + h_2 - z_0)}{h^*} \right),\end{aligned}\tag{2}$$

where $\sigma \eta_t$ represents Gaussian white noise of variance σ^2 . The values we are about to use are of an experiment where noise is neglected, hence $\sigma = 0$. Variable values used for the experiment are given in Table 1. The initial values Timmermann and Jin used for their study [TJ02, p.3-2] result in the spiraling orbit shown in Figure 3. Note that this orbit lives in a 3-dimensional space with the dimensions T_1, T_2 and h_1 .

α	$= 1/180 \text{ day}^{-1}$	r	$= 1/400 \text{ day}^{-1}$
T_{r0}	$= 16^\circ \text{ C}$	T_r	$= 29.5^\circ \text{ C}$
H_m	$= 50 \text{ m}$	H	$= 100 \text{ m}$
z_0	$= 75 \text{ m}$	h^*	$= 62 \text{ m}$
μ	$= 0.0026 \text{ K}^{-1} \text{ day}^{-1}$	$\mu b L / \beta$	$= 22 \text{ m K}^{-1}$
L	$= 15 \cdot 10^6 \text{ m}$	ε	$= 0.086$

Table 1: Variable values used for the experiment of Timmermann and Jin.

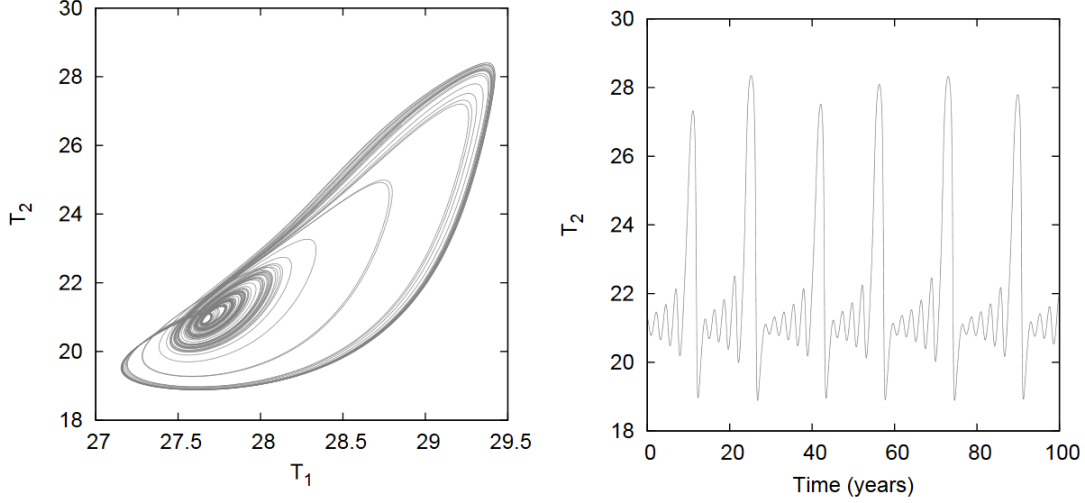


Figure 3: The resulting strange attractor, here T_1 and T_2 are the western and eastern equatorial Pacific sea surface temperatures.

Hypothesis and sub-questions

After creating and analyzing the toy model, we will discuss its usefulness with respect to the ENSO model. Ultimately, we will answer the following main question:

Can it be useful to employ a toy model over the ENSO model?

To be able to answer this question, we will consider the following sub-questions:

- *Can we simplify the ENSO model into a toy model that is able to simulate one variable accurately?*
- *How can we determine the difference in quality of results between the toy model and the ENSO model?*
- *What are the possible benefits of using such a toy model instead of the ENSO model?*
- *Aside from quality of results, are there other possible downsides on using a toy model instead of the ENSO model?*

After this introduction, we will proceed this first section with a short discussion on the El Niño phenomenon and a brief explanation on the ENSO model. In the second section, we will discuss all the preliminary theory on dynamical systems.

The third chapter is dedicated to deriving a toy model from the extreme values of the ENSO model. We also give methods to create similar models and state the models we will use for our analyses.

The analyses on the toy models will be in chapters four and five. In chapter four we will discuss basic properties of some of the models and in chapter five we will be focusing on a model with shifting top point. This analysis will show the differences in a continuous set of models.

2 Preparatory dynamical systems theory

As stated in the introduction, we will be considering models from dynamical systems theory. Therefore we start with providing some preliminary knowledge on this topic. Readers who are familiar with this discipline of mathematics can omit this chapter.

Notions described in this chapter are mostly adopted from the book of Devaney [D89]. Other references that are made use of are [HSD04] and [S94]. Backward asymptotic points for non-invertible maps are discussed in [R99].

2.1 General terminology

We start out with some general terminology used to distinguish different kinds of dynamical systems. Firstly, a **dynamical system** in general describes a point through time with a system of functions. The space in which the point is described is called the **phase space**. Let us discuss a more precise definition on dynamical systems.

Definition 1. *A dynamical system consists of a phase space X , a time set T and a function Φ . The function Φ is such that it maps a point in phase space $x_0 \in X$ and a point of time in the time set $t \in T$ to another (possibly the same) point in phase space $x \in X$. Hence $\Phi : X \times T \rightarrow X$ such that $\Phi(x_0, t) = x$.*

A dynamical system also follows the property that time can be added by repeating this function for that amount of time. That is,

$$\Phi(x_0, 0) = x \quad \& \quad \Phi(\Phi(x_0, t_1), t_2) = \Phi(x_0, t_1 + t_2), \quad (3)$$

for all $x_0 \in X$ and $t_1, t_2 \in T$.

The value x_0 is called the **initial condition**. Most dynamical systems only state their **prognostic equations**, since most dynamical systems are too involved to write down in one function depending on an initial condition and a point of time. We can distinguish two different types of dynamical systems, depending on their type of prognostic equations, namely discrete and continuous dynamical systems.

A **discrete dynamical system** has a prognostic equation that determines the next point using the current point. Time will progress in steps and hence the time set is a discrete sequence. A general example of a prognostic equation for a discrete dynamical system is

$$f(x) = \Phi(x, 1), \quad x \in X,$$

where $x = \Phi(x_0, t)$ for some $x_0 \in X$ and $t \in T \subseteq \mathbb{Z}$.

Continuous dynamical systems have differential equations as prognostic equations. Therefore time will be continuous for these systems. A general example of a prognostic equation for a continuous dynamical system is

$$f(x) = \frac{d}{dt}x, \quad x \in X,$$

where $x = \Phi(x_0, t)$ is its solution for some $x_0 \in X$ and $t \in T \subseteq \mathbb{R}$.

Solving this differential equation will directly result in the function of system (3). We also can plot the solutions of this system for some time interval $[t_{start}, t_{end}] \subseteq T$. The resulting curve is called a **trajectory** or a **solution curve**.

The **dimension** of a dynamical system denotes the number of variables that characterize the points in phase space. The ENSO model, for instance, is a 3-dimensional continuous dynamical system and its phase space has dimensions T_1, T_2 and h_1 . System (3) is only 1-dimensional and its phase space only has dimension X .

2.2 Basic concepts on mappings

The toy model will be a 1-dimensional discrete model and therefore discrete dynamical systems will be our primary focus. Recall that their prognostic equations determine the next point using the current point. These kind of equations are called mappings, or **maps** for short. The set of points used as input for the map is called the **domain** and the set of points used as output is called the **co-domain**. Because our maps will map points from one set back to the same set, its domain and co-domain will be the same. A map with this property, for a set X , can be denoted as $f : X \rightarrow X$.

Now consider a set $X \subseteq \mathbb{R}$ and a function $f : X \rightarrow X$. Also consider the subset $A \subset X$. The **image** of A is the set where all elements of A are mapped to by f and the **pre-image** of A is the set of elements that are mapped to A by f .

$$\begin{aligned} \text{Image of } A \text{ is} & \quad f(A) = \{f(a), a \in A\}. \\ \text{Pre-image of } A \text{ is} & \quad f^{-1}(A) = \{x \in X : f(x) \in A\}. \end{aligned}$$

It is important to stress that maps, by definition, map every point to another *single* point. Hence a point cannot be mapped to two different points, but it is possible that two points from the domain are mapped to the same point. However, when a map does map every single point to another unique point, then the map is called **one-to-one**. It is also not necessary for a map to map towards every single point of the co-domain. Maps that do map their domain towards every point of the co-domain are called **onto**. When maps are both one-to-one and onto, then they map every point of the domain to a unique point in the co-domain. These maps are called **bijective maps**.

In the latter case, the inverse of such a map is also a map and hence those maps are called **invertible**. We remark that the inverse of an invertible map is also bijective.

$$\text{Inverse of } f : A \rightarrow B \text{ is} \quad f^{-1} : B \rightarrow A, \quad \text{if } f \text{ is bijective.}$$

For non-invertible maps, it is possible to make them invertible. An non-onto map can be made bijective and hence invertible by shrinking its co-domain to the image of its domain. For non-one-to-one maps it is possible to consider **partial inverses** by cutting out a piece of the domain for which the map is one-to-one and by setting the co-domain as the image of that domain. Note that domains and co-domains differ for every different partial inverse of the same map.

Since we will use maps in a sequential manner, for example map $f : X \rightarrow X$, we occasionally we want to apply that map n times to some value $x \in X$ like $f(\dots f(x) \dots)$ or $f \circ \dots \circ f(x)$. For convenience, we can denote this operation as $f^n(x)$ and call it the **n -fold composition** of the map f . Furthermore, $f^n(x)$ is called the **n th iterate** of f from x .

2.3 Time series and Poincaré maps

For our toy model specifically, we point out that it will only consider closed intervals $[a, b] \subset \mathbb{R}$ and hence maps of our toy model will be of the form $f : [a, b] \rightarrow [a, b]$. Therefore we do not need to be more general than just generic closed intervals for all coming statements.

With that said, we will advance with the notion of time series. Like a continuous dynamical system has solution curves, a discrete dynamical system has time series. By gathering all iterates of a certain map, we create a sequence of iterates to which we can refer conform this definition.

Definition 2. Consider an interval $[a, b] \subseteq \mathbb{R}$ and a map $f : [a, b] \rightarrow [a, b]$.

The **time series** or **forward orbit** of the map f is the sequence we get from iterating an initial value $x_0 \in [a, b]$ with f . This will result in the sequence $(x_0, f(x_0), f^2(x_0), \dots)$ and we express this sequence as $O^+(x_0)$.

If f is invertible, it also has a **backward orbit** defined as $(x_0, f^{-1}(x_0), f^{-2}(x_0), \dots)$ and expressed as $O^-(x_0)$.

Another concept we explain, now that we have defined time series, are graphical analyses. A **graphical analysis** shows the progression of time series with the following procedure. Let us consider an interval $[a, b] \subseteq \mathbb{R}$, a map $f : [a, b] \rightarrow [a, b]$ and a point $x_0 \in [a, b]$ with time series $O^+(x_0)$.

1. We start by plotting the map in question f and the identity map.
2. We draw a vertical line or arrow from the identity map at the current iterate or initial value towards the map f .
3. Subsequently, we draw a horizontal line from the end of the previous line to the identity map. The value we end is the value of the next iterate.
4. Repeat the previous two step as much as desired.

Graphical analyses can depict the way dynamics of a map acts well and makes it more comprehensible. Examples of graphical analyses are shown in Figure 4. A concept that can comprehensibly show the dynamics of continuous dynamical systems is the Poincaré map.

Definition 3. Consider a continuous dynamical system Φ describing points γ in phase space $X \in \mathbb{R}^k$ through time interval $T \subseteq \mathbb{R}$ with initial condition $x_0 \in X$ such that

$$\Phi : (x_0, t) \rightarrow \gamma(t) \in X, \quad \forall t \in T.$$

Also consider a section $S \subset X$ chosen in such a way that γ intersects S after every orbit of γ . Let $(t_0, t_1, \dots) \subset T$ be the sequence of time points where γ intersects with S .

The function P that maps the point $\gamma(t_n)$ to the next point of intersection $\gamma(t_{n+1})$ is called the **Poincaré map**. The section S used for this Poincaré map is called a **Poincaré section**.

$$P : \gamma \cap S \rightarrow \gamma \cap S \text{ such that } p(\gamma(t_n)) = \gamma(t_{n+1}), \quad n \in \mathbb{N}.$$

Note that in the previous process, a continuous dynamical system is shaped into a discrete dynamical system. A Poincaré section selects a sequence of points from the solution curve for which its respective Poincaré map acts as the prognostic equation for the sequence $(\gamma(t_0), \gamma(t_1), \dots)$.

2.4 Fixed points, periodic points and their stability

From this section on, we will deal with concepts that actually analyse the dynamics of a discrete dynamical system. We start with some very basic yet important notions on dynamics of points with respect to the used map.

Definition 4. Consider an interval $[a, b] \subseteq \mathbb{R}$ and a map $f : [a, b] \rightarrow [a, b]$.

A **fixed point** of the map f is a point $x \in [a, b]$ such that f maps that point to itself, that is $f(x) = x$.

A **periodic point** of the map f is a point $x \in [a, b]$ such that there exists an $n \in \mathbb{N}$ for which f^n maps that point to itself, that is $f^n(x) = x$. That point is a periodic point of **period** n . The smallest $k \in \mathbb{N}$ for which $f^k(x) = x$ still holds is called the **prime period** of x .

Examples of fixed points can be found in Figure 7 & 9. Aside from a blue plot of a map, the identity function is also plotted in black. Points of the map that intersect with the identity line have the property that those values will remain the same after applying the map, which is exactly the definition of a fixed point. We remark that fixed points are periodic points with period 1.

Asymptoticity of points

Let us proceed with a notion relating dynamical characteristics to these fixed and periodic points.

Definition 5. Consider an interval $[a, b] \subseteq \mathbb{R}$ and a map $f : [a, b] \rightarrow [a, b]$ with a periodic point $p \in [a, b]$ of period n .

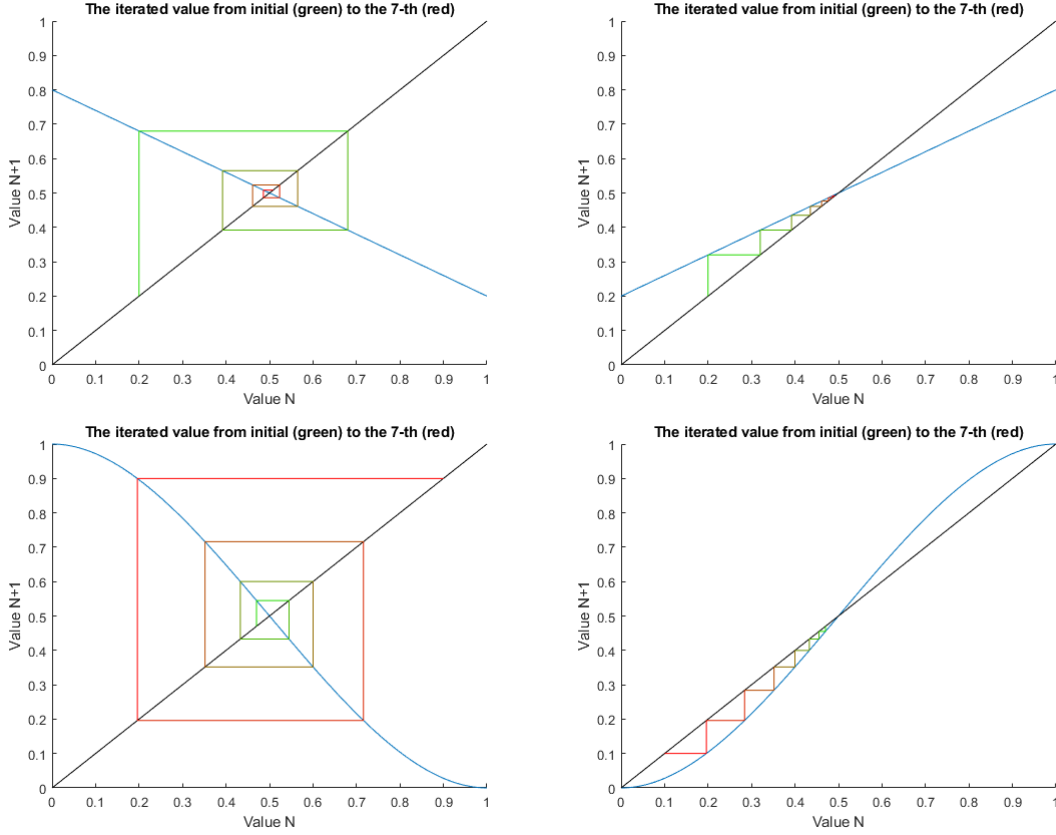


Figure 4: The iterations of two maps with 0.5 as stable fixed point and two maps with 0.5 as unstable fixed point. The initial value is at the start of the green line and is plotted vertically towards the map. The value after one iteration is made visual by plotting a horizontal line from that point towards the diagonal identity line. Repeating this process (from the green to the red lines) gives the trajectory of the iterations and thus shows the stability of the fixed point 0.5.

A point $x_0 \in [a, b]$, with time series $O^+(x_0) = \{x_0, x_1, \dots\}$ where $f(x_k) = x_{k+1}$, is **forward asymptotic** to p if x_{i-n} tends to p as i goes to infinity. That is $f^{i-n}(x_0) = x_{i-n} \rightarrow p$ as $i \rightarrow \infty$.

The **stable set** of p consists of all the points $x \in [a, b]$ that are forward asymptotic to p and is denoted as $W^s(p)$. Also, p is called an **attracting periodic point** in this case.

A point $x_0 \in [a, b]$ is **backward asymptotic** to p , even for a non-invertible map f , provided that there exists a time series $O^+(x_{-j-n}) = (x_{-j-n}, x_{1-j-n}, x_{2-j-n}, \dots)$ with $f(x_k) = x_{k+1}$ such that x_{-j-n} tends to p as j goes to infinity. That is $x_{-j-n} \rightarrow p$ as $j \rightarrow \infty$.

The **unstable set** of p consists of all the points $x \in [a, b]$ that are backward asymptotic to p and is denoted as $W^u(p)$. Also, p is called a **repelling periodic point** in this case.

Note that, although we could not define the backward orbit for non-invertible map, we could define how a map can be backward asymptotic to a periodic point. One does need to be careful, since it requires that we always can find a predecessor that is closer to the periodic point than the current point.

To improve our understanding of stable and unstable sets, we will illustrate them with a graphical analysis on four maps with a fixed point at 0.5 shown in Figure 4. The lines of the graphical analysis are drawn from green to red. The two upper graphs show the graphical analysis of two maps from the point 0.2. As we work through the process, we see that the iterates of the initial point tend towards the

fixed point 0.5. That means that its time series are within the stable set, which is $O^+(0.2) \subset W^s(0.5)$.

The two lower graphs show the same for two different maps from the point 0.47. As the process resolves, the iterates move from the fixed point 0.5. This indicates repelling behaviour, but does not prove that x_0 is backwards asymptotic to p . Therefore, we need to look at the pre-images. For invertible maps, like in Figure 5, we can determine the backward orbit and say $O^-(0.47) \subset W^u(0.5)$.

For non-invertible maps, on the other hand, we need to define a procedure that finds a point in every pre-image that is provably closer to the periodic point than the previous one. After this we can say that the resulting sequence lies within the unstable set, that is $O^+(f^{-j}(0.47))_{j \rightarrow \infty} \setminus O^+(f(0.47)) \subset W^u(0.5)$.

We will advance with more notions on periodic points. Before we introduce these concepts, we remark that the absolute value of the slope of the maps at 0.5 in the upper two graphs of Figure 4 is lower than 1 and the lower two have an absolute slope higher than 1. As one might point out, concepts concerning slopes of maps require that those maps are at least locally differentiable in the first place. A map is **differentiable** if its slope is well-defined for every point of its domain.

Definition 6. Consider an interval $[a, b] \subseteq \mathbb{R}$ and a differentiable map $f : [a, b] \rightarrow [a, b]$ with a periodic point $p \in [a, b]$ of period n .

The **multiplier** of the periodic point p is given by the slope of the n -fold composition of the map f at point p , that is $(f^n)'(p)$.

The periodic point p is **hyperbolic periodic point** if the absolute value of its multiplier is not equal to 1.

Stable and unstable sets

We saw that the maps of Figure 4 with a fixed point that has a multiplier with an absolute value lower than 1 had time series within the stable set and the maps with a fixed point with absolute multiplier higher than 1 had time series within the unstable set. And indeed, this is no coincidence, as the upcoming proposition will show.

Proposition 1. Consider an interval $[a, b] \subseteq \mathbb{R}$ and a differentiable map $f : X \rightarrow X$ with a periodic point $p \in X$ of period n such that its multiplier has $|(f^n)'(p)| < 1$. Then there exists an open interval S about p that is within its stable set $W^s(p)$.

Proof of this theorem is mostly based on the proof in the book of Devaney [D89, p.25].

Proof. Since the map f has $|(f^n)'(p)| < 1$, we have that f^n is at least once differentiable at p . Thus there exists $\epsilon > 0$ such that there exists a number A with $|(f^n)'(x)| < A < 1$ for all $x \in [p - \epsilon, p + \epsilon] \cap X$.

According to the **mean value theorem**, there exists an a , with $a \in [x, p]$ if $x < p$ or $a \in [p, x]$ if $x > p$, such that $f^n(x) - f^n(p) = (f^n)'(a)(x - p)$. This theorem can be found in [D89, p.10].

Since a is in between x and p , we know that $a \in [p - \epsilon, p + \epsilon] \cap X$ and thus $|(f^n)'(a)| < A < 1$. Therefore $|f^n(x) - f^n(p)| < A|x - p|$. From this follows that

$$|f^n(x) - p| = |f^n(x) - f^n(p)| < A|x - p| < |x - p|.$$

Note that $f^n(x)$ is lying closer to p than x did. Now let us consider $f^{i \cdot n}(x)$ for $i \in \mathbb{N}$.

$$|f^{i \cdot n}(x) - p| = |f^{i \cdot n}(x) - f^n(p)| < A|f^{(i-1)n}(x) - p| < \dots < A^i|x - p|.$$

And therefore, since $A < 1$, we have that $f^{i \cdot n}(x) \rightarrow p$ as $i \rightarrow \infty$. Together with Definition 5, we know there exists $\epsilon > 0$ such that all $x \in [p - \epsilon, p + \epsilon]$ are forward asymptotic to p and thus are contained within the stable set. And therefore there exists an open interval S about p such that $S \subset W^s(p)$. \square

With this proposition, we can directly show the existence of a stable set around a periodic point only by showing that the absolute value of its multiplier is lower than 1. An opposite proposition concerning periodic points can also be proven.

Proposition 2. Consider an interval $[a, b] \subseteq \mathbb{R}$ and a differentiable map $f : [a, b] \rightarrow [a, b]$ with a periodic point $p \in [a, b]$ of period n such that its multiplier has $|(f^n)'(p)| > 1$. Then there exists an open interval U about p that is within its unstable set $W^u(p)$.

Proof. The proof is mostly the same as the previous proof and therefore we will use shorter notation. $\exists \epsilon > 0$ such that $|(f^n)'(x)| > A > 1, \forall x \in [p - \epsilon, p + \epsilon] \cap [a, b]$.

Then, with the mean value theorem, $\exists a \in [p - \epsilon, p + \epsilon] \cap [a, b]$ such that

$$|f^n(x) - p| = |(f^n)'(a)| \cdot |x - p| > A|x - p| > |x - p|.$$

Now let us fix x for a moment and try to find the wanted predecessor and call it x_1 . Assume x_1 exists within $[p - \epsilon, p + \epsilon] \cap [a, b]$, then $\exists b \in [p - \epsilon, p + \epsilon] \cap [a, b]$ between x_1 and p such that

$$x - p = f^n(x_1) - p = (f^n)'(b)(x_1 - p) > A(x_1 - p),$$

according to the mean value theorem. Next, $\epsilon > |x - p| > A|x_1 - p|$ results in $x_1 \in [p - \epsilon/A, p + \epsilon/A] \cap [a, b]$, proving that x_1 indeed exists in $[p - \epsilon, p + \epsilon] \cap [a, b]$. Repeating this process for the prior point of x_1 , say x_2 , we get $x_2 \in [p - \epsilon/A^2, p + \epsilon/A^2] \cap [a, b]$.

Let x_i be the i th point prior to x we choose by repeating the previous process. This point has $x_i \in [p - \epsilon/A^i, p + \epsilon/A^i] \cap [a, b]$ and has existing time series $O^+(x_i)$. Hence x is backward asymptotic since $x_i \rightarrow p$ as $i \rightarrow \infty$ by Definition 5.

This holds $\forall x \in [p - \epsilon, p + \epsilon] \cap [a, b]$ and thus there exists an open interval U about p such that $U \subset W^u(p)$. \square

Now that we have proven their existence, let us name these open intervals which are described in the previous two propositions.

Definition 7. The open interval S described by Proposition 1 is called the **local stable set** and is denoted as W_{loc}^s .

The open interval U described by Proposition 2 is called the **local unstable set** and is denoted as W_{loc}^u .

Lyapunov exponents

After we have defined stable and unstable sets for periodic points, we introduce another notion on stability using slopes. This time, we are looking at the average slope of points in time series. The idea is that points attract each other if their absolute slope is smaller than 1 and repel if its larger than 1. Therefore, an average slope would indicate if points of time series attract or repel each other in general.

Definition 8. Consider an interval $[a, b] \subseteq \mathbb{R}$, a map $f : [a, b] \rightarrow [a, b]$ and time series $O^+(x_0) = \{x_0, x_1, \dots\}$ where $x_n \in [a, b]$ for all $n \in \mathbb{N}$.

We define the **time- n Lyapunov exponent** λ_n of x_0 as the average of the logarithm of the slope of the first n iterates. Note that by the chain rule, this is equivalent to the logarithm of the slope of the n -fold composition of the map at x_0 .

$$\lambda_n(x_0) = \frac{1}{n} \sum_{i=0}^{n-1} \log |f'(x_i)| = \frac{1}{n} \log \left| \prod_{i=0}^{n-1} f'(x_i) \right| = \frac{1}{n} \log |f^n'(x_0)|.$$

We define the **Lyapunov exponent** λ as the average of the logarithm of the slope of each iterate of the time series $O^+(x_0)$. This corresponds to the time- n Lyapunov exponent λ_n of x_0 in the case that n goes to infinity.

$$\lambda = \lim_{n \rightarrow \infty} \frac{1}{n} \sum_{i=0}^{n-1} \log |f'(x_i)|.$$

Note that the logarithm of the slope of an attracting point will be negative and that of a repelling point positive. Hence a negative Lyapunov exponent will indicate time series with stable behaviour while a positive Lyapunov exponent indicates one with unstable behaviour.

Most of the time, a Lyapunov exponent cannot be calculated analytically and requires numerical computation. As a consequence, it is not possible to determine the true Lyapunov exponent. Instead, a time- n Lyapunov exponent has to do. One can imagine that the results of a time- n Lyapunov exponents depends greatly on the chosen initial value x_0 . Indeed, every bounded time series has different iterates and all iterates have different slopes. Of course, the significance of this issue decreases to the extent of which the number of iterates to evaluate increases. Still, one has to take note of this fact when computing Lyapunov exponent numerically.

2.5 Snap-back repellers

In the previous section, we discussed notions on local stable and unstable sets. However, it can be proven that a dynamical systems can be unstable everywhere. These systems are called chaotic systems.

Before we dive in, we discuss two notions we will use in this section. Firstly, we will use the notion of a neighbourhood. A **neighbourhood** is typically an open set, ball or interval of any positive diameter $r > 0$ around some point x_0 . This is often denoted as $B_r(x_0)$. When we consider points inside a neighbourhood around x_0 when r is not already defined, then for every $r > 0$, these points x that is inside the neighbourhood around x_0 the distance between x and x_0 is smaller then any positive , then there exists a diameter $r > 0$ such that the distance between x and x_0 is less than r . A **continuously differentiable** map is a function with a continuous derivative.

Proving that a dynamical system is chaotic, or proving chaos in general, is very difficult. The first problem that arises is the question what chaos or a chaotic system actually is. This fundamental question gave rise to different definitions for chaos. The most popular definition is given by Devaney [D89, p.50].

Definition 9. Consider a set X and a map $f : X \rightarrow X$. The map f is said to be **chaotic in the sense of Devaney** on X if

1. f has sensitive dependence on initial conditions;
2. f is topologically transitive. i.e. $\forall U, V \subseteq X, \exists k > 0$ such that $f^k(U) \cap V \neq \emptyset$;
3. periodic points are dense in X .

Although we now have a widely accepted definition for chaos, it is still a lot of work to prove chaos for dynamical systems. Marotto came up with an easy way to prove chaos in 1978. This is done by finding a snap-back repeller. This method, however, was not presented without flaws. This was pointed out by Chen, Hsu and Zhou in 1998. In the paper of Shi and Chen from 2004, an improvement was made on the needed definition of an expanding point [SC04, p.557].

Definition 10. Consider the set X and a map $f : X \rightarrow X$ and a fixed point $p \in X$. The fixed point p is an **expanding fixed point** of f in the closed interval $\bar{B}_r(p) \subseteq X$ for some $r > 0$ if $\exists \lambda > 1$ such that $|f(x) - f(y)| \geq \lambda|x - y|, \forall x, y \in \bar{B}_r(p)$.

If p is an interior point of $f(B_r(p))$, then p is called a **regular expanding fixed point**

We remark that this condition is met for 1-dimensional systems if the absolute value of the slope of $f(x)$ for points x inside the closed interval $\bar{B}_r(p)$ is higher than 1. We proceed with the definition of a snap-back repeller given in the article of Marotto [M78, p.203].

Definition 11. Consider the set X , a map $f : X \rightarrow X$ and a fixed point $p \in X$. The fixed point p is a **snap-back repeller** if

1. p is an expanding point of f in $B_r(p) \subseteq X$.
2. $\exists x_0 \in B_r(p)$ with $x_0 \neq p$ such that $\exists k \geq 2$ with $f^k(x_0) = p$.

Furthermore, p is called a **non-degenerate snap-back repeller** of f if

3. $f(B_r(p))$ is open;
4. $\exists \delta_0 > 0$ such that $B_{\delta_0}(x_0) \subset B_r(p)$;
5. p is an interior point of $f^k(B_\delta(x_0))$ for each positive $\delta \leq \delta_0$.

To prove that snap-back repellers indeed prove chaos, Marotto first made his own definition of chaos, which can be found in the same article we just mentioned.

Definition 12. Consider a set X and a map $f : X \rightarrow X$. The map f is said to be **chaotic in the sense of Marotto** on X if it possesses a snap-back repeller. That is, there exists

1. a positive integer N such that, $\forall n \geq N$, $\exists p \in X$ that is a periodic point of f of period n ;
2. an uncountable set S containing no periodic points of f such that
 - (a) $f(S) \subset S$;
 - (b) $\forall x, y \in S$ with $x \neq y$ we have $\limsup_{k \rightarrow \infty} |f^k(x) - f^k(y)| > 0$;
 - (c) $\forall x \in S$ and any periodic point p of f we have $\limsup_{k \rightarrow \infty} |f^k(x) - f^k(p)| > 0$;
3. an uncountable set $S_0 \subset S$ such that $\forall x, y \in S_0$ we have $\liminf_{k \rightarrow \infty} |f^k(x) - f^k(y)| = 0$.

This list of criteria turned out to be not sufficient for results Marotto claimed. Therefore, two additional criteria were given such that the results of Definition 12 hold [SC05, p.234].

Definition 12 (continued). Furthermore, we assume that

4. $\forall \epsilon > 0$ the map f is continuously differentiable inside a neighbourhood of p and all eigenvalues of $Df(p)$ have absolute values higher than 1. This implies that there exists a neighbourhood around p which is an unstable set of p .
5. p is a snap-back repeller of f with $f^k(x_0) = p$ such that f is continuously differentiable in the neighbourhoods of x_j and have $\det Df(x_j) \neq 0, \forall j \in \{0, 1, \dots, k-1\}$ where $x_j = f(x_{j-1})$.

The fourth criterion was initially put in the definition of an expanding point. The fifth criterion takes into account that the map f at all the points of the forward orbit of x_0 until p need to be continuously differentiable. This means that the map f does not need to be entirely continuously differentiable, which we will need for our toy model.

The same theorem we referred to for the previous two criteria also shows us in which case f is also proven to be chaotic in the sense of Devaney.

Theorem 1. If f satisfies all five conditions of Definition 12, then, for each neighbourhood U of p , there exist a positive integer $l > k$ and a Cantor set $\Lambda \subset U$ such that $f^l : \Lambda \rightarrow \Lambda$ is chaotic on Λ in the sense of Devaney.

In 2005, the following theorem is shown as a remark in an article written by Shi and Yu [SY05]. For our research, this theorem will be the one of importance to prove chaos.

Theorem 2. A regular non-degenerate snap-back repeller implies chaos in the sense of Devaney.

2.6 Bifurcation theory

This section will be devoted to bifurcation theory. Our main source for this section will be again the book of Devaney, specifically starting at [D89, p.80]. We will also use material from [BT09]. This theory focuses on families of 1-dimensional systems that depend smoothly on the parameter μ . **Smooth** functions are functions that are infinitely differentiable with respect to a given variable. The family of 1-dimensional systems can be described by a two dimensional function with domain $X \times M \subseteq \mathbf{R}^2$,

$$G(x, \mu) = f_\mu(x),$$

where f_μ is a 1-dimensional dynamical system if $\mu \in M$ is fixed. Devaney only considers smooth 1-dimensional systems. We, on the other hand, start with systems which are not necessarily smooth, but only continuous.

Bifurcation theory on families of smooth and non-smooth maps

Actual bifurcation theory can only be applied on smooth functions. This does not withhold us from evaluating families of non-smooth maps. Families of maps, smooth or not, can be evaluated with the use of a bifurcation diagram.

A **bifurcation diagram** is a product of a numerical experiment in which we determine the set of values $X_\mu \subseteq X$ that occur after $k \in \{m+1, \dots, n\} \subseteq \mathbb{N}$ iterations of the map for some initial value $x_0 \in X$. This is repeated for each value of $\mu \in M$.

$$X_\mu = \{x \in X \mid f_\mu^k(x_0) = x, \forall k \in \{m+1, \dots, n\} \subset \mathbb{N}\}$$

Note that the values of the set X_μ depend on the initial value $x_0 \in X$. Ideally, we want to reduce the dependence of the initial value to get a good representation of the values we can expect in general. This dependence can be reduced in two ways.

Firstly, we can ignore the first m iterations of f on the initial value $x_0 \in X$, for each $\mu \in M$. These ignored values are called the **transient**. By eliminating the transient, we ignore the values that are highly dependent on the initial condition. The larger the transient, the lesser X_μ depends on x_0 . The second way is to let X_μ be a large set. This increases the variation of possible values of X_μ and thus they become less dependent on the initial value.

3 The toy model and its map approximations

The dynamics of the ENSO model correspond well with El Niño phenomenon. But getting specific information requires to solve a system of 3 differential equations, which takes a lot of computing capacity. If one is only interested in one specific variable, then a 1 dimensional model would be more efficient. This leads to the first sub-question of our research:

Can we simplify the ENSO model into a toy model that is able to simulate one variable accurately?

3.1 Derivation of the toy model

Now that we have a clear goal, we need a way to achieve this. For this purpose, we will roughly pursue the following steps:

- Find 'an easy to model' pattern within the results of the ENSO model;
- Pick a smaller set of variables that could be predicted separately from the other variables;
- Use these results to create functions which are only dependent on the chosen set of variables.

Evaluating the results of the ENSO model

The results of the ENSO model, i.e. its solution curve with respect to the used initial conditions, is shown in the graphs of Figure 3. The first graph shows the solution curve projected onto the (T_1, T_2) -plane orbiting with rotations of varying length. Its progression through time goes clock-wise. The second graph shows T_2 as a function of time. These orbits of the solution curve in the left graph of Figure 3 appear here as oscillations. From this graph, a notable pattern is visible. The temperature shows a repeating pattern of an oscillation with an increasing amplitude.

As mentioned in the previous chapter, the phase space of the ENSO model has 3 dimensions, namely T_1, T_2 and h_1 . But one might expect, due to the seemingly predictable value of T_2 with respect to time, that it potentially can be modeled with only its previous values and thus by neglecting T_1 and h_1 .

Choosing restrictions that lead to the toy model

To support our expectations on this pattern of T_2 over time, we take a look at the behaviour of the local maxima of T_2 with respect to time. Hence, with respect to the solution curve of the ENSO model, we consider the times t_n , such that

$$\frac{dT_2}{dt}(t_n) = 0 \quad \& \quad \frac{d^2T_2}{dt^2}(t_n) > 0. \quad (4)$$

These criteria with respect to the solution curve of the ENSO model are marked with a blue line in Figure 5. This line embodies a 2-dimensional surface within the 3-dimensional phase space of the ENSO model. We can use this surface as a Poincaré section to create a Poincaré map (Definition 3), with which we can create a function to determine every subsequent local maximum. Hopefully, this will strongly depend on T_2 values of previous local maxima.

As said after the definition of a Poincaré map, we can reduce this continuous model to a discrete model. Hence, we are able to create a sequence of time points from a continuous time interval that relates to the time series of the Poincaré map. This produces a less involved dynamical system which has the Poincaré section as its phase space.

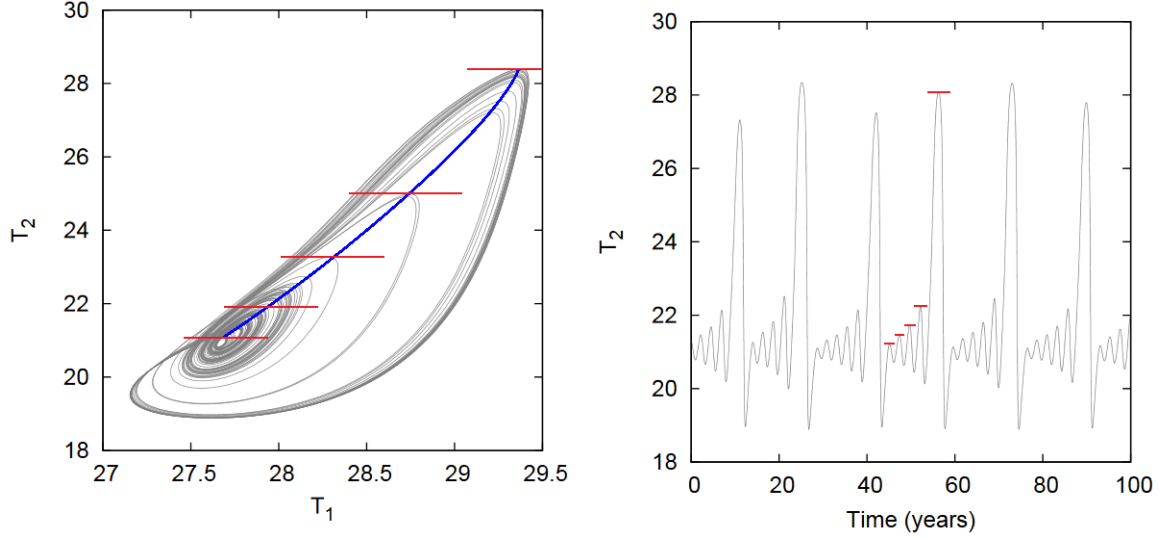


Figure 5: The strange attractor of the ENSO model. The restrictions of equations (4) result in the blue line, which embodies a surface within the 3 dimensions of the ENSO model (T_1, T_2, h_1) . To point out that intersections with the solution curve are indeed local maxima with respect to T_2 , some points highlighted with a red tangent line.

Explicitly describing the Poincaré section

The solution curve of the ENSO model passes the surface of equations (4) once after every orbit and thus these equations meet the criteria for a Poincaré section. Using the differential equations of the ENSO model (1) and equations (4), we can express this Poincaré section explicitly.

$$\begin{aligned}
 0 &= \frac{dT_2}{dt}(t_n) = -\alpha(T_2(t_n) - T_r(t_n)) - \frac{w(T_2(t_n) - T_{sub}(t_n))}{H_m}, \\
 0 &< \frac{d^2T_2}{dt^2}(t_n) = -\alpha\left(\frac{dT_2}{dt}(t_n) - \frac{dT_r}{dt}(t_n)\right) - \frac{w}{H_m}\left(\frac{dT_2}{dt}(t_n) - \frac{dT_{sub}}{dt}(t_n)\right) \\
 &= \alpha\frac{dT_r}{dt}(t_n) + \frac{w}{H_m}\frac{dT_{sub}}{dt}(t_n).
 \end{aligned}$$

A discrete function with only 1 dimension

Recall that we can relate any local maximum on the solution curve of the ENSO model to the next with the previously mentioned Poincaré map. Let us denote this map as P . Furthermore, let M_n be the value of T_2 at the n -th local maximum after the initial value M_0 . The Poincaré map P is a function from the Poincaré section $S \subset \mathbb{R}^3$ to itself, that is $P : S \rightarrow S$. Therefore we have

$$P(M_n, T_1(t_n), h_1(t_n)) = (M_{n+1}, T_1(t_{n+1}), h_1(t_{n+1})), \quad \forall n \in \mathbb{N}.$$

Our goal, however, is to only focus on the values M_n and see if those values relate to each other.

$$f : S \rightarrow S, \quad \text{such that} \quad f(M_n) = M_{n+1}, \quad \forall n \in \mathbb{N}. \quad (5)$$

The question is whether this function makes sense. Is the pattern so strong that can we say something about M_{n+1} by knowing M_n ? To answer that question, we make a plot of each local maxima against the next one. The result of that plot is given in the first graph of Figure 6.

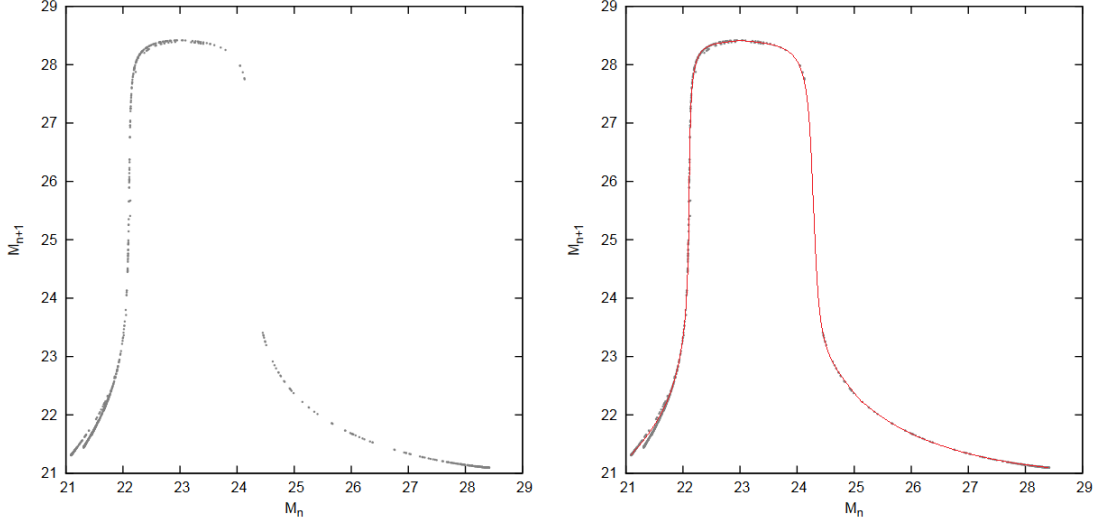


Figure 6: The sequence of local maxima $(M_n)_{n \in (1, 1000)}$. Here, M_{n+1} is plotted versus M_n .

We see that the idea of a function f , as given in equation (5), indeed makes sense. The pattern we have seen in the oscillation of T_2 in Figure 3 induces a strong relation between M_n and M_{n+1} that seems to lie on a single smooth curve, which is shown with a red curve in the second graph. Therefore, it seems that we indeed can simulate a variable of the ENSO model accurately. To be able to properly answer this question, we need to see if the toy model also has a corresponding behaviour to the ENSO model. Hence we first look further into the map f by creating approximations and evaluating their dynamics.

Before we proceed with these approximations, we first rescale and rename the variables M_n . We rescale them in such a way that the lowest temperature has value 0, the highest temperature has value 1 and hence they all are within unit interval $[0, 1]$. Furthermore, we are naming the values x_n in stead of M_n after re-scaling. In the next sections we will be discussing approximations of this map $f : [0, 1] \rightarrow [0, 1]$.

3.2 A piece-wise linear approximation

The easiest way to approximate map f is with a piece-wise linear approximation. Therefore the first approximation will be of that kind. Such an approximation can be made by choosing k points $(a_1, b_1), \dots, (a_k, b_k)$ on the graph of f , where $f(a_n) = b_n$ and

$$0 = a_1 < \dots < a_k = 1.$$

The $k - 1$ lines between these points define the map $f_{k-1} : [0, 1] \rightarrow [0, 1]$.

$$f_{k-1}(x) = b_i + \frac{b_{i+1} - b_i}{a_{i+1} - a_i}(x - a_i) \quad \text{with} \quad x \in [a_i, a_{i+1}], \quad \forall i \in [1, \dots, k-1]. \quad (6)$$

The 7-piece linear approximation

The first approximation we will consider is a 7-piece linear map with the following set of points:

$$(a_1, b_1) = (0, 0.03), \quad (a_2, b_2) = (0.1, 0.2), \quad (a_3, b_3) = (0.15, 0.94), \quad (a_4, b_4) = (0.27, 1), \\ (a_5, b_5) = (0.41, 0.94), \quad (a_6, b_6) = (0.46, 0.32), \quad (a_7, b_7) = (0.6, 0.12), \quad (a_8, b_8) = (1, 0).$$

The graph of rescaled local maximum temperatures versus the next one (x_n, x_{n+1}) and the piece-wise linear approximation f_7 is shown in Figure 7.

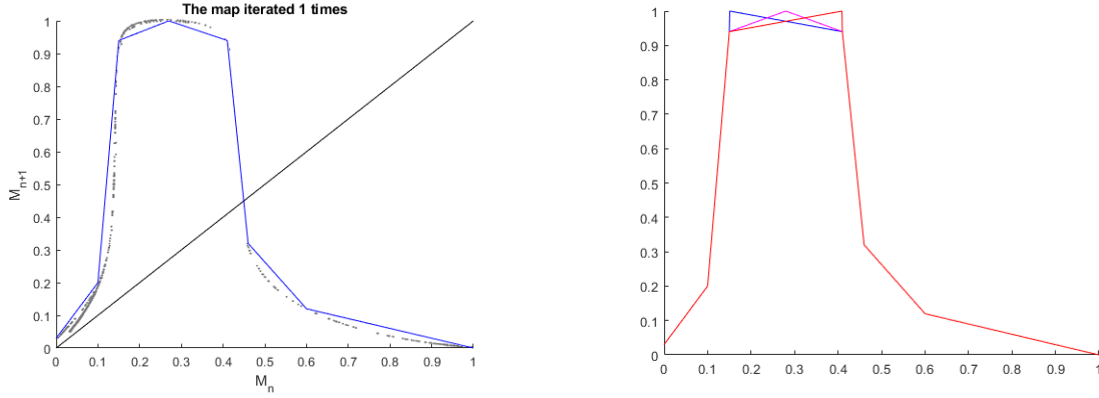


Figure 7: The rescaled local maximum temperatures (x_n), the linear approximation f_7 and the linear approximation with shifting point $f_{7\mu}$ with $\mu = 0.151, 0.28, 0.409$.

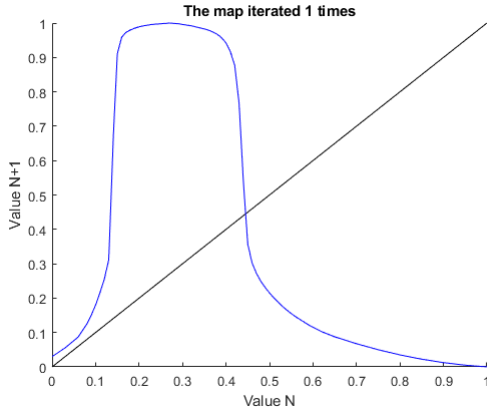


Figure 8: The rescaled local maximum temperatures and its 50-piece linear approximation f_{50} .

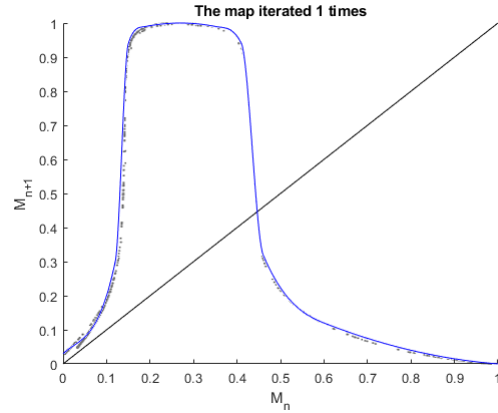


Figure 9: The rescaled local maximum temperatures and its piece-wise third degree polynomial approximation f_p .

The 50-piece linear approximation

For the sake of creating a map which is able to replicate data of the ENSO model more accurately, a second linear approximation is made. This version uses 50 lines and is partly designed using reverse engineering, by comparing prospective results with real life El Niño phenomenon occurrences. This resulted in the map f_{50} , shown in Figure 8.

3.3 A piece-wise linear approximation with a shifting point

A variation on the piece-wise linear approximation is one with a shifting point. With this, we consider a continuous set of linear maps where a component of one point varies, hence $a_n = \mu$ or $b_n = \mu$. We let its value vary such that $a_{n-1} < \mu < a_{n+1}$. Such a set is an example of a family of map as described in the beginning of Section 2.6.

By evaluating a set of map approximations with a shifting point, we can see if the dynamics of a map change when we shift this point. This enables us to investigate to what extent small differences in maps can change the dynamics of a system.

The k -piece linear approximation with shifting point $f_{k\mu}$ we will use will be the linear map using

the same set of points as the first approximation f_7 , except for the top point. For the top point, we will let the value $a_4 = \mu$ vary such that $0.15 < \mu < 0.41$. We remark that b_4 will not be changed when we use this approximation. Three maps of the set $f_{7\mu}$ are shown in the right picture of Figure 7

3.4 A piece-wise third degree polynomial approximation

Another possible approximation to the map f is with a piece-wise third order polynomial approximation. This approximation can be made by choosing k points with their slopes $(a_1, b_1, s_1), \dots, (a_k, b_k, s_k)$ on the graph f , where $f(a_n) = b_n, s_n \in \mathbb{R}$ and

$$0 = a_1 < \dots < a_k = 1,$$

and by defining the map $f_p : [0, 1] \rightarrow [0, 1]$ as

$$f_p(x) = \alpha_i x^3 + \beta_i x^2 + \gamma_i x + \delta_i \quad \text{with} \quad x \in [a_i, a_{i+1}], \quad \forall i \in [1, \dots, k-1],$$

where coefficients $[\alpha_i, \beta_i, \gamma_i, \delta_i] \in \mathbb{R}^4$ are such that

$$\begin{bmatrix} a_i^3 & a_i^2 & a_i & 1 \\ a_{i+1}^3 & a_{i+1}^2 & a_{i+1} & 1 \\ 3a_i^2 & 2a_i & 1 & 0 \\ 3a_{i+1}^2 & 2a_{i+1} & 1 & 0 \end{bmatrix} \begin{bmatrix} \alpha_i \\ \beta_i \\ \gamma_i \\ \delta_i \end{bmatrix} = \begin{bmatrix} b_i \\ b_{i+1} \\ s_i \\ s_{i+1} \end{bmatrix}$$

since $f_p(a_i) = b_i, f_p(a_{i+1}) = b_{i+1}, f'_p(a_i) = s_i$ and $f'_p(a_{i+1}) = s_{i+1}$.

One should be careful with this approximation since it is very likely to cross the boundaries of unit interval or not having unit interval as its entire range when applying this approximation carelessly. An appropriate set of points that keeps this remark into account can be:

$$\begin{aligned} (a_1, b_1, s_1) &= (0, 0.03, 1.2), & (a_2, b_2, s_2) &= (0.12, 0.3, 6), & (a_3, b_3, s_3) &= (0.15, 0.94, 4), \\ (a_4, b_4, s_4) &= (0.19, 0.99, 0.25), & (a_5, b_5, s_5) &= (0.35, 0.99, -0.25), & (a_6, b_6, s_6) &= (0.41, 0.94, -2.6), \\ (a_7, b_7, s_7) &= (0.46, 0.32, -3), & (a_8, b_8, s_8) &= (0.6, 0.12, -0.5), & (a_9, b_9, s_9) &= (1, 0, -0.1). \end{aligned}$$

This set of intervals is appropriate because the fourth interval with slopes is chosen in such a way that the result is a parabola that exactly touches the value 1 at $x = 0.27$.

The graph of rescaled local maximum temperatures and the piece-wise polynomial approximation f_p is shown in Figure 9.

4 Basic dynamics of the toy model

Now that we have derived our toy model, we will analyse their dynamics. In the upcoming sections, we will analyse the following dynamical properties of the approximations we have made in the previous chapter. We will explore the following properties considering the piece-wise linear and polynomial approximation, mentioned in the previous chapter.

- Time series of iterated values;
- Occurring El Niño periods;
- The fixed and periodic points;
- Stable and unstable sets of the time series.

As the observant reader may notice, we have not yet discussed El Niño periods. Therefore, let us discuss it now.

Definition 13. Consider an interval $[a, b] \subseteq \mathbb{R}$, a map $f : [a, b] \rightarrow [a, b]$ and a point $x_0 \in [a, b]$ with time series $O^+(x_0)$.

Iterates $x_n \in O^+(x_0) = \{x_0, x_1, \dots\}$ such that $x_n > x_{n+1}$ where $n \in \mathbb{N}$ are denoted as an **El Niño point**.

Now consider two consecutive El Niño points $x_a, x_b \in O^+(x_0)$, where $a < b$. The number of iterations it takes to get from x_a to x_b is the period of the El Niño point x_b , or **El Niño period** for short, and is equal to $b - a$.

Before we elaborate on basic properties on specific approximations separately, we will discuss what they have in common.

4.1 General properties of the approximations

We begin our analysis with a quick comparison between the time series' pattern of the linear approximation and the local maxima of the ENSO model, found in Figure 11 & 3 respectively, to see if we succeeded in reproducing the oscillating pattern. Indeed, the repeated pattern of increasing local maximum values of the ENSO model is also visible with our toy model.

As the title of this section suggests, the other maps will have some overlapping characteristics. This is not necessarily because the chosen approximations are close enough to the local maxima of the ENSO model, but due to its global shape with respect to its fixed point.

El Niño maps and similar behaviour

We start with how this "global shape" will look like. We will denote four characteristics and name these approximations El Niño maps.

Definition 14. A continuous prognostic map $f : [0, 1] \rightarrow [0, 1]$ is called an **El Niño map** if it meets these four requirements:

1. f is onto.
2. f has exactly one fixed point $p \in (0, 1)$.
3. There exists exactly one point $q \in (0, p)$ such that $f(q) = p$.
4. There is a point t such that $f(x)$ for $x \in (0, t)$ is increasing (i.e. if $x_1 > x_2$, then $f(x_1) > f(x_2)$) and $f(x)$ for $x \in (t, 1)$ is decreasing (i.e. if $x_1 > x_2$, then $f(x_1) < f(x_2)$).

To illustrate this definition, we will check if the linear approximations of Section 3.2 meets its requirements. We check that $f_7 : [0, 1] \rightarrow [0, 1]$ is indeed onto since it is continuous and has points $f_7(0.27) = 1$ & $f_7(1) = 0$, meaning that its co-domain equals its domain. Next, f_7 indeed has one fixed point around $0.45 \in (0, 1)$ and a point $f(q) = p$ round $0.12 \in (0, p)$. The last requirement is easily tested by checking its construction in Section 3.2. From the definition of El Niño maps, we can directly state some consequences.

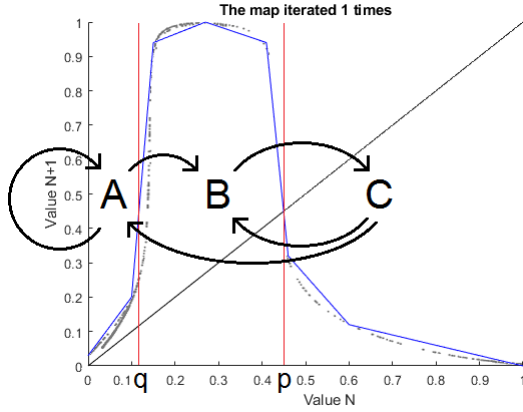


Figure 10: The approximation f_7 and its domain divided in intervals $A = [0, q)$, $B = (q, p)$ and $C = (p, 1]$. The arrows indicate to which interval a point inside some interval will be mapped.

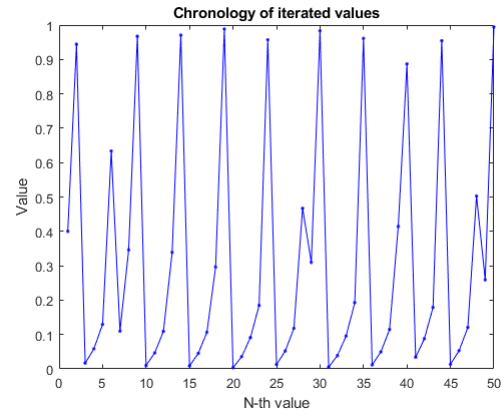


Figure 11: The first 49 iterations of the linear approximation f_7 with initial value 0.4.

Corollary 1. *An El Niño map $f : [0, 1] \rightarrow [0, 1]$ with fixed point p and $q \neq p$ such that $f(q) = p$ has the following direct results:*

1. *The point such that $f(t) = 1$ has $t \in (q, p)$;*
2. *$f(1) = 0$.*

With these results we can have 3 separate cases for all $x \in [0, 1] \setminus \{q, p\}$:

3. *If $x < q$, then $x < f(x) < p$;*
4. *If $q < x < p$, then $p < f(x)$;*
5. *If $p < x$, then $f(x) < p$.*

From the latter, we get the following consequences:

6. *x is an El Niño point if and only if $p < x$;*
7. *every El Niño point has at least period 2.*

The proof of each item of this corollary can be found in Appendix. By using item 3 to 5 of Corollary 1 we divide the domain of f into three intervals $A = [0, q)$, $B = (q, p)$ and $C = (p, 1]$ and evaluate how iterates of each interval develop, as is represented with arrows in Figure 10.

This can be done for any El Niño map due to their similar behaviour. The sequence of non-El Niño points prior to an El Niño point $x_n \in (p, 1]$ of period a will always be $x_{n-a} < \dots < x_{n-1} < x_n$ such that $\{x_{n-a}, \dots, x_{n-2}\} \subset [0, q)$ and $x_{n-1} \in (q, p)$.

Partial inverse of approximations restricted to the non-decreasing part of the map

The third requirement of El Niño maps $f : [0, 1] \rightarrow [0, 1]$ with fixed point p is that they need to have a point $q \in (0, p)$ such that $f(q) = p$. For the linear approximation we have $q \approx 0.12$. We see that the results of the ENSO model have $q \approx 0.14$.

We also can determine for how many iterations a value will stay within interval $[0, q)$ if we go back further. We know that, by definition, El Niño maps are not invertible. But we do not necessarily have to consider the inverse of the whole map, since we are only looking to points smaller than q . Note that, by requirement 4 of Definition 14, there must exist a point t such that $f(x)$ is increasing for all $x \in (0, t)$ and by item 1 of Corollary 1, $q < t$, implying that $f(x)$ is one-to-one for at least $x \in [0, q]$.

To be more exact, the map is one-to-one up to the point that $f(x) = 1$ and its one-to-one after that point.

Therefore, we can consider left and right partial inverses of El Niño maps f , denoted as f_l^{-1} and f_r^{-1} respectively. The left partial inverse has domain $[f(0), 1]$ and co-domain $[0, t]$ and the right partial inverse has domain $[0, 1]$ and co-domain $[t, 1]$. We will illustrate these inverses with the example of a k -piece linear approximation f_k of Section 3.2 by inverting equation (6). The result, a general formula and the left and right inverse of the 7-piece linear approximation, are given underneath.

$$f_k^{-1}(x) = a_i + \frac{a_{i+1} - a_i}{b_{i+1} - b_i}(x - b_i), \quad x \in [b_i, b_{i+1}], \quad \forall i \in [1, \dots, k-1], \quad \text{for } b_1 \leq x \leq 1, \quad (7)$$

$$f_{7,l}^{-1}(x) = \begin{cases} \frac{x-0.03}{1.7}, & \text{if } 0.03 \leq x < 0.2, \\ \frac{x-0.2}{14.8} + 0.1, & \text{if } 0.2 \leq x < 0.94, \\ \frac{x-0.94}{0.5} + 0.15, & \text{if } 0.94 \leq x \leq 1, \end{cases} \quad \text{for } 0.03 \leq x \leq 1. \quad (8)$$

$$f_{7,r}^{-1}(x) = \begin{cases} \frac{7(1-x)}{3} + 0.27, & \text{if } 1 \geq x > 0.94, \\ \frac{0.94-x}{12.4} + 0.41, & \text{if } 0.94 \geq x > 0.32, \\ \frac{7(0.32-x)}{10} + 0.46, & \text{if } 0.32 \geq x > 0.12, \\ \frac{0.12-x}{0.3} + 0.6, & \text{if } 0.12 \geq x \geq 0, \end{cases} \quad \text{for } 0.03 \leq x \leq 1. \quad (9)$$

We emphasize that left and right partial inversions are indeed possible for every El Niño map. Due to the simplicity of the linear approximation f_7 and its straightforward partial inverse, we take it as our example. A general representation of a partial inverse restricted to the non-decreasing part of some El Niño map f would be $f^{-1} : [f(0), 1] \rightarrow [0, f^{-1}(1)]$.

Predicting periods of El Niño points with El Niño characteristics

Recall that intervals A and B from Figure 10 are separated by the value q such that $f(q) = p$. Note that $q = f_l^{-1}(p)$, where f_l^{-1} denotes the left partial inverse of the map f . Points x such that $f_l^{-1}(p) < x < p$ appear in interval B and will be mapped by f to an El Niño point. Points $x < f_l^{-1}(p)$ appear in interval A , which means that f would map them towards a point lower than the fixed point p . Therefore, the point $f_l^{-1}(p)$ separates the points $x \in [0, p]$ into a set of points that become an El Niño point after 1 iteration and a set that become an El Niño point after 2 or more iterations.

We can repeat this process for the latter set of points $x \in [0, f_l^{-1}(p)]$. In this fashion, we can find out which points will be an El Niño point after exactly 2 iterations. These points are such that $f_l^{-2}(p) < x < f_l^{-1}(p)$ since the right inequality indicates that x is not an El Niño point after 1 iteration, while the left inequality indicates that x will be an El Niño point within 2 iterations. This pattern can be repeated until iterates of the partial inverse f_l^{-1} is mapped outside its domain. For the linear approximation f_7 , that occurs when $f_{7,l}^{1-a}(p) \in [0.03, f(0.03))$ and in general this is when $f_l^{1-a}(p) \in [f(0), f^2(0))$ for some $a > 1 \in \mathbb{N}$.

We can check that iterations of partial inverses of the fixed point p divide the domain $[0, 1]$ into a open intervals and half-open intervals with one end-point that is either 0 or 1. For all El Niño points $x_n \in O^+(x_0)$, $x_n > p$, each of these open intervals have elements such that $f_l^{-k}(p) < x_{n-k} < f_l^{1-k}(p)$ for $k \in \{1, 2, \dots, a-1\}$, where $f^k(x_{n-k}) = x_n$ and $a \in \mathbb{N}$ is such that $f_l^{1-a}(p) \in [0, f(0))$. We call $a+1$ the **maximum period of El Niño points** for the map f , since it is the largest El Niño period possible. This number differs per El Niño map.

We also name the sequence $(f_l^{1-a}(p), f_l^{2-a}(p), \dots, f_l^{-1}(p), p)$ which separates open and half-open sets of points that need the same number of map iterations to become an El Niño point. We define this sequence as the **El Niño characteristics**. Lastly, we remark that the number of El Niño characteristics is one less than the maximum orbit. A visualization of all notions we just defined is given in Figure 12.

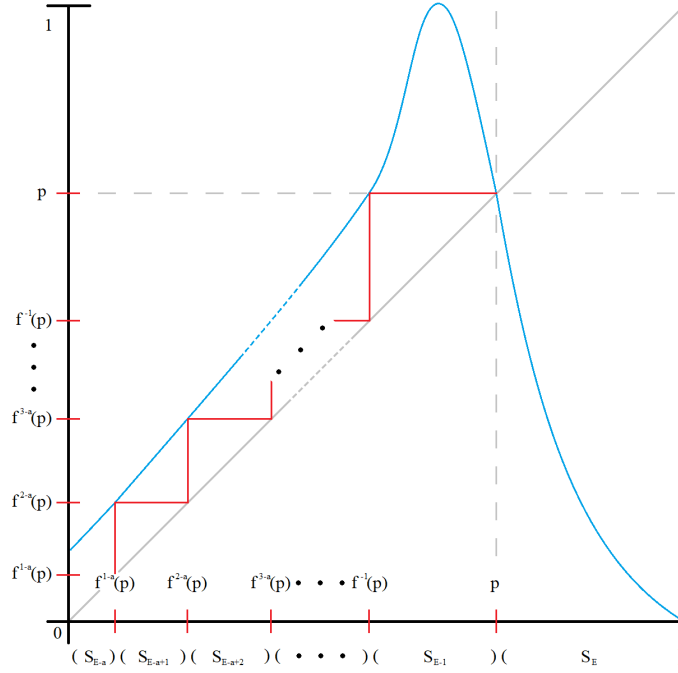


Figure 12: An illustration of an El Niño map f with a graphical analysis towards its fixed point p . The El Niño characteristics are also denoted by $f^k(p)$ where $k \in \{1 - a, 2 - a, \dots, 0\}$ and the sets of points S_{E-i} where $i \in \{a, a - 1, \dots, 0\}$ indicates the number of iterations it takes for a point in that set to become an El Niño point. Note that $a + 1$ is the maximum period of f .

4.2 Basic properties of linear approximations

For the properties of linear approximations, we will mainly focus on the 7-piece approximation f_7 . Thereafter, we will quickly state some properties of the 50-piece approximation f_{50} and discuss the differences between them.

In the previous section, we already analysed some specific properties of the linear approximation f_7 while we investigated general properties of approximations. For this section, we will solely focus on the linear approximation and its features. The results in this section are based on numerical computation concerning the four points stated in the beginning of this chapter. Hence, the values of time series, El Niño periods, fixed and periodic points and stable and unstable sets will be discussed.

Time series and El Niño periods

We start with two histograms, given in Figure 13 and 14, showing the occurrences of values and El Niño periods. For the values, we see that values around 0, 0.1 and 1 occur the most. If we look closer, we see five peaks, around 0.01, 0.05, 0.12, 0.19 and 0.96. When we look at Figure 7, we see that the most grey point also clutter around these values and thus they indeed seem to occur frequently.

Some of this behaviour can be explained with the histogram of the El Niño periods. Every possible El Niño period smaller or equal than 5 occurs. El Niño periods of 5 occur the most, which could explain the five peaks we have seen in the histogram of the values. Probably, sequences with values close to the sequence (0.19, 0.96, 0.01, 0.05, 0.12) occur frequently for the map f_l , since they roughly follow each other.

To make these deductions more explicit, we start with the El Niño characteristics of f_7 . These can be calculated with the inverse, which was already given explicitly in equation 8, and are given as (0.012, 0.051, 0.117, 0.450). This confirms that the maximum El Niño period is indeed equal to 5.

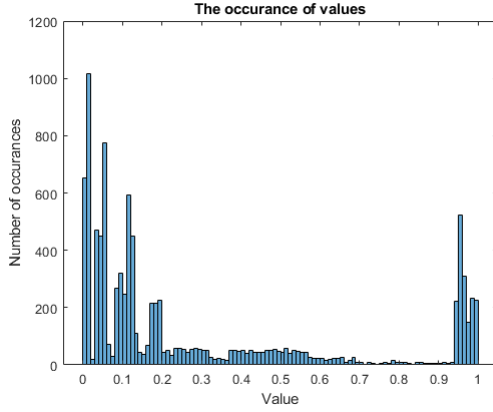


Figure 13: A histogram of 100 segments of the first 10,000 iterations of f_7 with initial value 0.4.

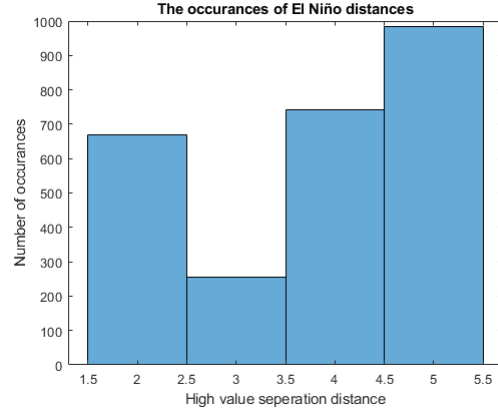


Figure 14: A histogram showing the frequency of periods of El Niño points of f_7 .

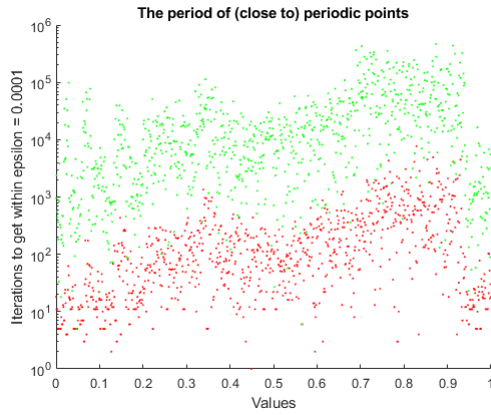


Figure 15: The number of iterations it takes such that an initial point comes back within 0.01 (red) and 0.0001 (green) from the initial value.

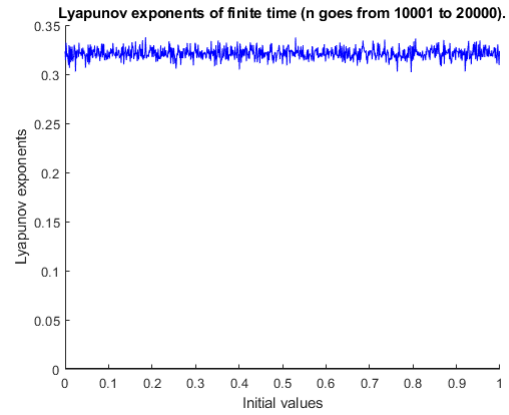


Figure 16: The time-20,000 Lyapunov exponent of 10,000 initial values of the linear approximation.

Periodic points

Determining all periodic points numerically is impossible, since their periods can become infinitely large. But we can give an indication of how many there are by investigating whether a point comes close to its initial value. Figure 15 shows how many iterations it takes for a point to return within a certain epsilon. For the linear approximation, all 1000 evenly spaced points inside the domain return within 0.0001 after 500,000 iterations.

The previous result gives us the suspicion that, after many iterations, we can get arbitrarily close to any initial value, indicating that periodic points might be dense. This is one of the requirements for chaos according to Devaney, as seen in Definition 9.

Lyapunov exponent

We continue by estimating the Lyapunov exponent of this map. We estimate the Lyapunov exponent by using a time-20,000 Lyapunov exponent with a transient of 10,000 iterations. The results, illustrated in Figure 16, show that the approximation has a time-20,000 Lyapunov exponent between 0.3 and 0.33, indicating repelling behaviour for initial values along the whole domain.

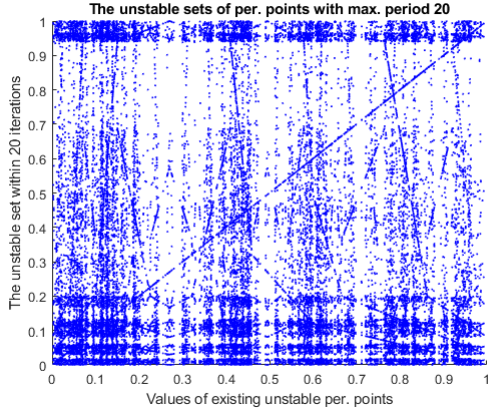


Figure 17: All periodic point with prime period less than or equal to 20. Here, for each periodic point, the forward orbit of p for 20 iterations displayed. Hence, the time series for each periodic orbit is shown for each periodic point.

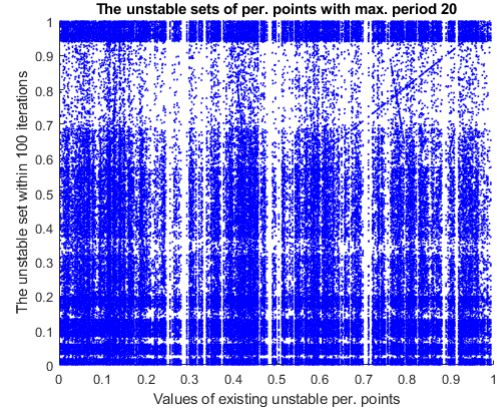


Figure 18: All periodic point with prime period less than or equal to 20. Here, for each periodic point, the forward orbit of $p + 0.01$ is shown for 100 iterations. Hence, the unstable set of 100 iterations is shown for every periodic point p .

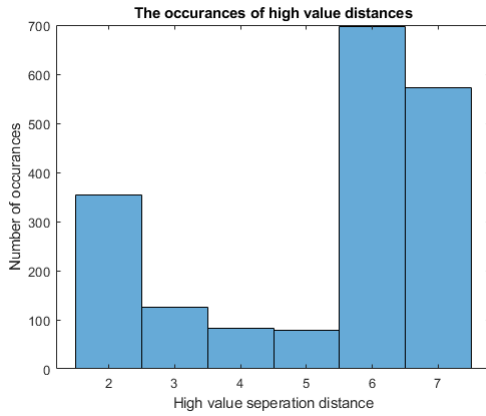


Figure 19: A histogram of occurring El Niño periods in the first 10,000 iterations of f_{50} .

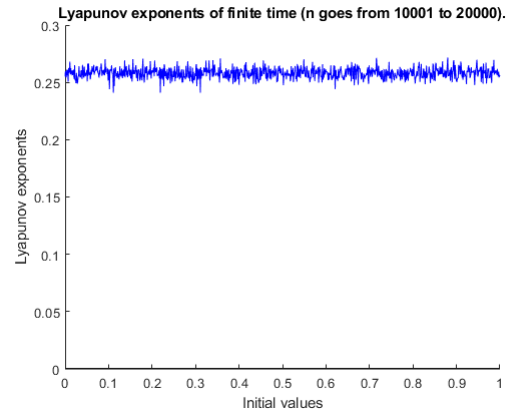


Figure 20: The time-20,000 Lyapunov exponent of 10,000 initial values of the map f_{50} .

Unstable sets

We have seen that repelling behaviour runs along the whole domain. But we have not seen whether the unstable sets of periodic points consist of the whole domain or just a subset of the domain.

To find this out, we first evaluate a set of periodic point. Figure 17 shows for every periodic point, the value of the x -axis, its time series on the y -axis for each $p = x$. By making a small perturbation in the value of p , in this case $\epsilon = 0.01$, we will get an unstable set. In Figure 18, we have chosen to let the unstable set have 100 elements. When we further increase that number, the whole interval will be blue. Thus for every unstable periodic point, the whole domain seems to be the unstable set for every periodic orbit.

The 50-piece Linear approximation

Aside from the map f_7 , we also consider the linear approximation f_{50} . Since these maps are essentially of the same kind, we will not analyse it extensively. A property of both of these maps is that they

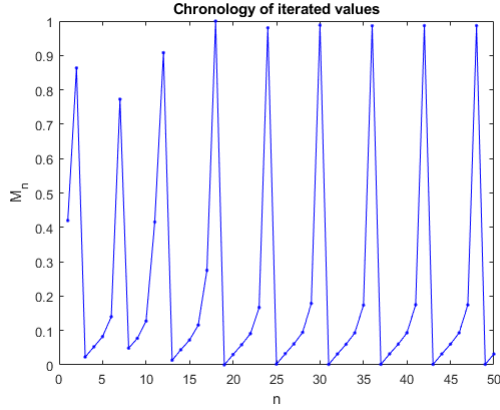


Figure 21: The first 49 iterations of f_p with initial value 0.42.

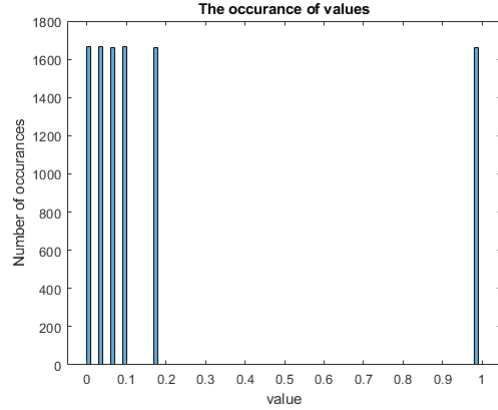


Figure 22: A histogram of 100 segments of the first 10,000 iterates of f_p with initial value 0.42.

both behave chaotically. The main difference between f_7 and f_{50} is the difference in the maximum orbit of El Niño points. Figure 19 shows the its maximum orbit of El Niño points is 7, which is two iterations more than f_7 .

Some slight differences is the fixed point, which is equal to $\frac{8457}{19000} \approx 0.4451$, and its Lyapunov exponent, as is shown in Figure 20. The Lyapunov exponent of f_7 jumps around the value 0.32, while the value of the Lyapunov exponent of f_{50} is around 0.26.

4.3 Basic properties of the polynomial approximation

As done with the linear approximation, we will engage in analysing basic properties of the polynomial approximation. To start off, we take note that there are a number of appearance differences between the polynomial and linear approximation. The most notable are that f_p is smooth, while f_l is obviously not smooth, and this approximation looks like a better fit to the points of the ENSO model, which can be seen in Figure 9.

Time series and El Niño periods

We start our analysis with its time series and a histogram of the values of its time series, given in Figure 21 & 22. At first glance, the time series seem similar to the one of the linear approximation, but the histogram shows something odd: Six values occur equally often. The histogram of El Niño periods would look equally monotonous: Only period six occurs.

The reason for these appearances in values can be seen in figure 21. In the beginning, the iterates vary in value, but quickly stabilize. After iterate 25, the periods already seem identical. This behaviour leads us directly to the next section.

Periodic points, stable sets and unstable sets

Before we focus on the six values that stabilize quickly, we start with the periodic points of the map. As we have done for the linear approximation, we have evaluated the first 500,000 iterates of 1000 equally separated values of the domain of the polynomial approximation and the results are given in Figure 23.

This plot shows far less iterates that reappear in the neighbourhood of the initial point. Also notable is the fact that the highest number of iterations for which an iterate to be inside a neighbourhood of 0.01 of an initial value is 32, considering the 500,000 iterates we have evaluated. Only 3 of the 1000

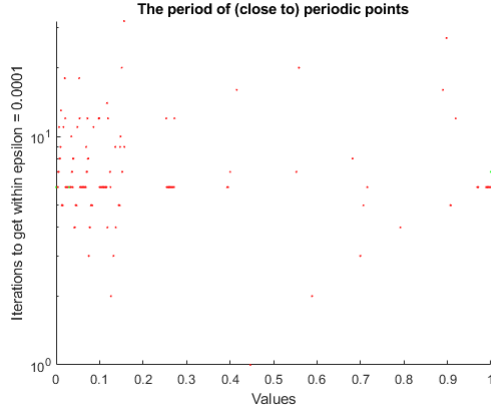


Figure 23: The number of iterations it takes such that an initial point comes back within 0.01 (red) and 0.0001 (green) from the initial value.

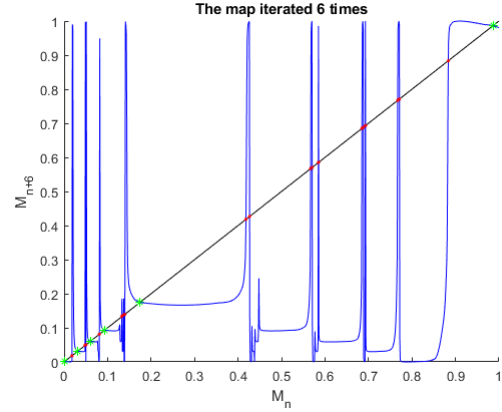


Figure 24: The 6-fold composition of f_p . Periodic points are shown in red when multiplier m has $|m| \geq 1$ and with a green star when $|m| < 1$.

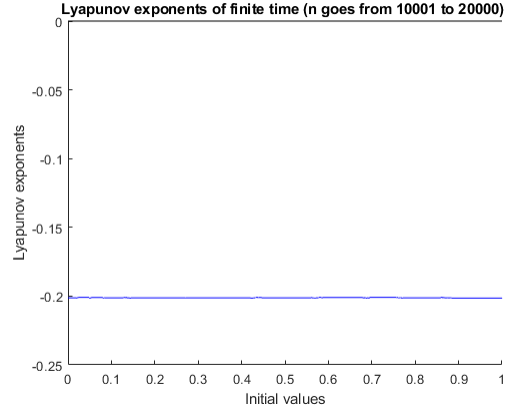


Figure 25: The time-20,000 Lyapunov exponent of 10,000 initial values of the linear approximation.

points have an iterate within 0.0001 of its initial value, opposed to the linear approximation, for which all points had an iterate within 0.0001 of its initial value.

Let us go back to the six stabilizing values. This nature coincides with the notion of forward asymptoticness. The initial value tend to six attracting periodic points. When we evaluate the time series of every initial value of the domain, the same behaviour will occur. This explains the plot of Figure 23, since all values will quickly tend towards these attracting periodic points and hence it suggests that the whole domain will be the stable set of the polynomial approximation.

Before we illustrate this with the Lyapunov exponent, we show the 6-fold composition of the polynomial approximation in Figure 24. Note that there are periodic points with multipliers less than 1, which implies that there exists stable sets by Proposition 1.

In Figure 25, we have the results of the time-20,000 Lyapunov exponent of every initial value. Indeed, its value for every initial point is slightly lower than -0.2 and almost looks like a constant line, emphasizing its stable nature.

5 Bifurcation of the linear approximation with shifting top point

We proceed our analysis with the toy model given by the linear approximation with shifting top point we described in Section 3.3. This continuous set of maps only changes in its top section $f(x) > 0.94$, due to a changing top point $f(\mu) = 1$ where $\mu \in (0.15, 0.41)$. Note that the map discussed in Section 4.2 belongs to this set and can be obtained if we choose $\mu = 0.27$.

We start our analysis with a bifurcation diagram of this family shown in Figure 26. We see that for values of μ higher than approximately 0.186, and hence also for $\mu = 0.27$, all values between 0 and 1 seem to occur. Values of μ lower than approximately 0.186, on the other hand, show a significant smaller set of values for X_μ and a lower Lyapunov exponent, shown in Figure 28 and 29. More specifically, we see five branches of occurring values. One of these branches is shown in more detail in Figure 27. We will treat every change in dynamics of these branches.

To treat these dynamics we will use close-ups of the 5, 10, 15 and 30-fold composition of the linear approximation. The full scale versions in the case that $\mu = .1677$ are shown in Figure 30 to 33.

5.1 Period tripling

The first change of behaviour happens at precisely $\mu = 0.1677$. At every branch of Figure 26, the five lines tend monotonically towards the points 0, 0.03, 0.081, 0.1677 and 1 is split into three lines at this point. Period tripling is a rare phenomenon and cannot be explained using conventional bifurcation theory, which is focused on smooth maps. Any piece-wise linear El Niño map is not smooth and therefore this kind of strange behaviour can occur.

The number 0.1677 is no coincidence since the time series starting from $x = 0$ is 0, 0.03, 0.081, 0.1677 for any μ . When $\mu = 0.1677$, the next iterate will be 1, which is followed by 0, completing the circle. Since the same type of period cannot exist after $\mu = 1677$, it evolves from a periodic point of period 5 to one of period 15.

A close-up of the 5-fold composition of the map at $\mu = 0.167$, $\mu = 0.1677$ and $\mu = 0.1685$, shown in Figure 34, shows that the fixed point around $p = 0.1677$ becomes unstable. As the map changes continuously, the kink moves past the black identity line and the slope of the fixed point becomes a steep slope.

For all maps with $\mu \in [0.151, 0.1677)$, their fifth composition has a stable fixed point and thus they have a stable periodic point of period 5. The time series of these periodic points are the stable fixed points of the 5-fold composition and these values are the start of each of the five branches of the bifurcation diagram of Figure 26.

After $\mu = 0.1677$, we have period tripling at every branch, indicating periodic points of period 15. Therefore we will examine the 15-fold composition of the map. A close-up of the 15-fold composition of the map with $\mu = 0.167$, shown in Figure 35, also shows a stable fixed point at the same location. This was to be expected, since periodic points of period 5 are also periodic period 15. The difference between period 5 and 15 is, however, that after $\mu = 0.1677$, spikes with three stable fixed points emerge after the kink with the latter case. These stable points emerge in every branch, pinpointing all 15 values of the time series of the periodic points of period 15. As the μ progresses, these spikes grow, as can be seen in Figures 36.

5.2 Chaotic behaviour inside branches

When we follow the 15-fold composition of the map for increasing values of μ , we also see that spikes move away from the identity line and eventually, they do not intersect anymore. This happens in the last plot of Figure 36, after $\mu = 0.174$. At this point, the spikes lose their three stable fixed points. As expected, this also happens with the 30-fold composition in Figure 39. The three stable fixed points are shown and disappear in the second plot, again, after $\mu = 0.174$.

The difference between the 30 and 15-fold composition is the number of unstable fixed points. It turns out that, between the highest and lowest fixed points, there are many unstable fixed points. The range of these fixed points grow with the amplitude of the spikes. The result of what we observe is again visible in the bifurcation diagram, Figure 27. The stable period 15 periodic points disappear around $\mu = 0.174$ and the many unstable periodic points between the highest and lowest stable periodic points take over in every branch.

In Figure 29, we see the Lyapunov exponent jump at that point to a positive but still relatively low value. The periodic points are not stable, but only a small set of points occur in the bifurcation diagram. This does not mean that periodic points do not exist outside these branches, as can be seen in Figure 39 and 40.

The range of values inside the branches of the bifurcation diagram

The reason they do not occur in the bifurcation diagram is very interesting to research and probably has something to do with basins of attraction. But with a look at the forward orbits, we already can see points stay locked up inside these branches of Figure 26. Let us take $\mu = 0.175, 0.182$ and look at forward orbits with $x_0(0)$ and $x_{\text{end}}(1)$ for this experiment.

$$\begin{aligned}\mu = 0.175, \quad O^+(x_0 = 0) &= (0, 0.03, 0.081, 0.1677, 0.9825, 0.0053, 0.0389, 0.0962, 0.1935, 0.9953), \\ O^+(x_9 = 1) &= (0.0070, 0.0419, 0.1013, 0.2191, 0.9902, 0.0029, 0.0350, 0.0895, 0.1821, 1); \\ \mu = 0.182, \quad O^+(x_0 = 0) &= (0, 0.03, 0.081, 0.1677, 0.9732, 0.0080, 0.0437, 0.1042, 0.2628, 0.9787), \\ O^+(x_9 = 1) &= (0.0055, 0.0393, 0.0969, 0.1947, 0.9950, 0.0015, 0.0326, 0.0854, 0.1751, 1).\end{aligned}$$

Our aim is to proof that values of time series stay within the branches. First we consider the forward orbits for $\mu = 0.175$. We start with $x_0 = 0$ and continue towards $x_4 = 0.9825$. For any value of μ , the line segments are increasing and therefore, for a some range, these will be the lowest values of each branch. That is, if $x_0 = 0$ is contained inside the lowest branch.

For the next value of the forward orbit, we note that the next line segment of the map is decreasing. Therefore, since $x_4 = 0.9825$ is considered to be the lowest value of the highest branch, the next value $x_5 = 0.0053$ must be the highest of the lowest branch. When we proceed, we come across $x_8 = 0.1935$. Note that for every interval $[x_0, x_5], [x_1, x_6], [x_2, x_7]$ and $[x_3, x_8]$ fits inside a single linear segment of the map.

For the next interval, we need to note that $x_8 = 0.1935 > \mu = 0.175$ and $x_9 = 0.9953 > x_4 = 0.9825$ and thus values that are mapped from the interval $[x_3, x_8]$ will be within $[0.9825, 1]$. Looking back with the forward orbit $O^+(x_9 = 1)$, we see that these values are indeed within the given intervals. Therefore, the values of sets of branches indeed stay inside the branches.

For the case where $\mu = 0.182$, we see a great increase of range of values in the bifurcation diagram of Figure 27, specifically starting from the value 0.2. Looking at the end of branch interval $[x_3, x_8] = [0.1677, 0.2628]$, we see a great increase in its value. This is due to reaching the end of a line segment of the map at 0.2, as can be seen in Figure 7. The slope increases, explaining the great increase of range of values with relatively low density. Still we have that $x_9 = 0.9787 > x_4 = 0.9732$ and thus the same procedure to find ranges of value inside the branch applies. When this was not the case, we choose x_9 to be the lowest number of the highest branch.

5.3 Period 10 solutions and the wall of chaos

After the preciously mentioned sets of chaos, when we proceed with increasing the value μ , we see that the values start to converge towards two lines of values, ultimately resulting in seemingly stable periodic point. Since this again occurs at every branch, its period must be 10.

To show these periodic points indeed become stable, we will look to a close-up of the 10-fold composition of the linear approximation with shifting top point $\mu \in [0.177, 0.187]$, shown in Figure 37 and 38. Indeed, we see that stable periodic points occur at values $\mu = 0.185$ and $\mu = 0.186$. The slope of the two fixed points change continuously and are -1.133, -1.018, -0.993 and -0.970 for μ equals 0.18, 0.184, 0.185 and 0.186 respectively, confirming their attracting nature.

While the slope of these fixed points becomes more and more mellow, the fixed point moves towards a kink and eventually reaches a segment with a very steep slope. This happens between $\mu = 0.186$ and $\mu = 0.187$ as can be seen in Figure 38. After this point, the values of the bifurcation diagram occur over the whole domain. This change in behaviour is again very interesting to research and probably has to do with basins of attraction.

As a summary of the events discussed in the current and previous section, one can consider the development of the 30-fold composition for $\mu = [0.174, 0.187]$. In Figure 39, we see the stable periodic points of period 15 in the first plot and the steep spikes at values above 0.2 in the third plot. This progresses in Figure 40, where periodic point of period 10 become stable in the second plot and disappear in the third plot.

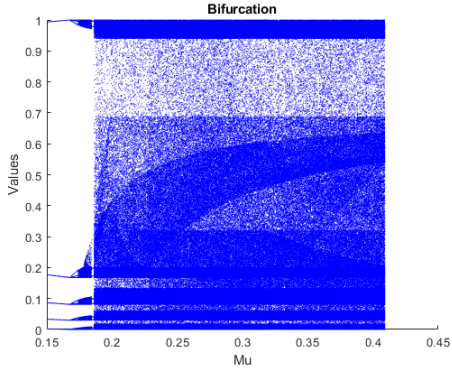


Figure 26: The bifurcation diagram of the linear approximation with shifting top point $\mu \in [0.151, 0.409]$.

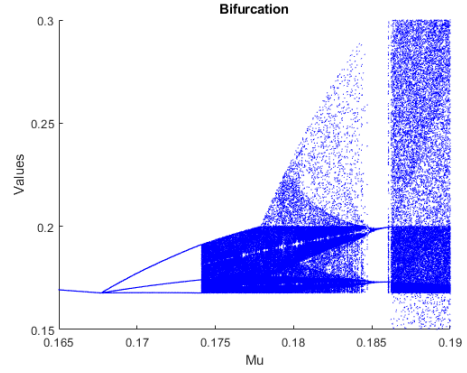


Figure 27: The bifurcation diagram of the linear approximation with shifting top point $\mu \in [0.165, 0.19]$.

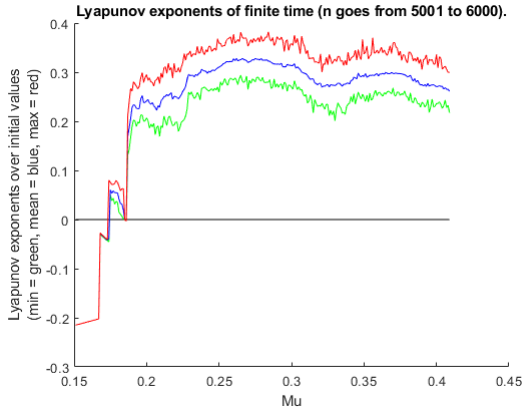


Figure 28: The Lyapunov exponent range for the linear approximation with shifting top point $\mu \in [0.151, 0.409]$.

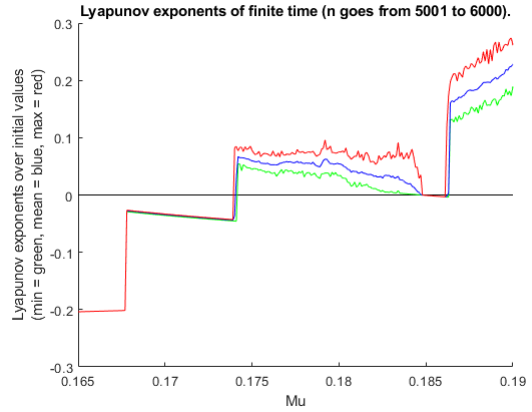


Figure 29: The Lyapunov exponent range for the linear approximation with shifting top point $\mu \in [0.165, 0.19]$.

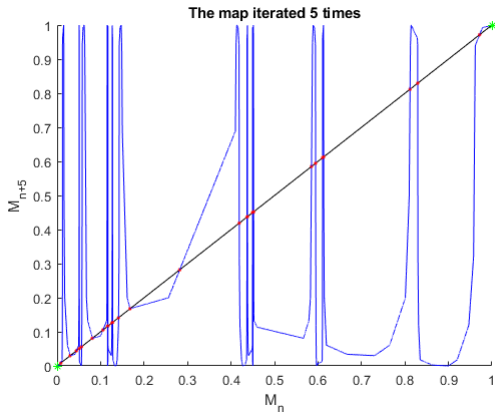


Figure 30: The 5-fold composition of the linear approximation with top point set to $\mu = .1677$

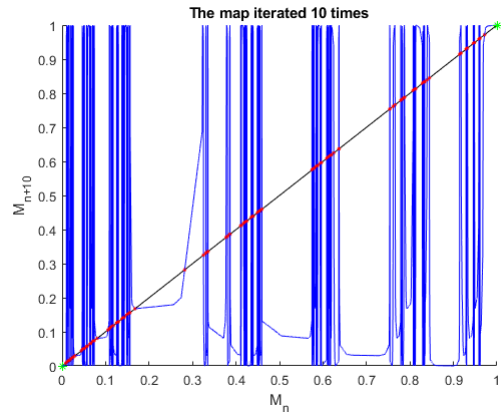


Figure 31: The 10-fold composition of the linear approximation with top point set to $\mu = .1677$

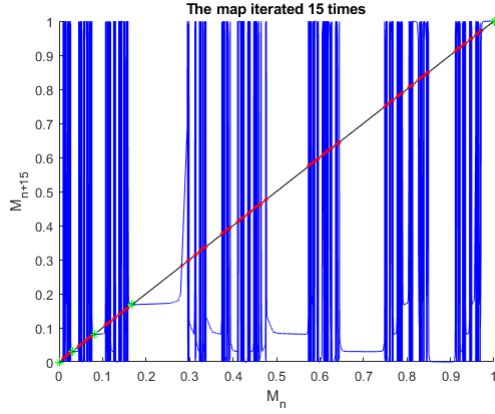


Figure 32: The 15-fold composition of the linear approximation with top point set to $\mu = .1677$

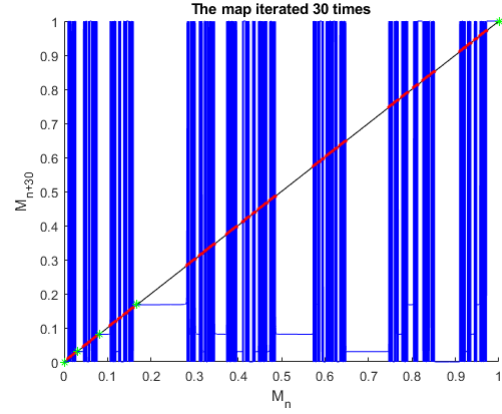


Figure 33: The 30-fold composition of the linear approximation with top point set to $\mu = .1677$

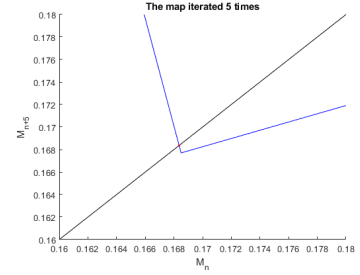
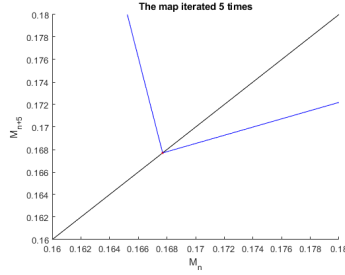
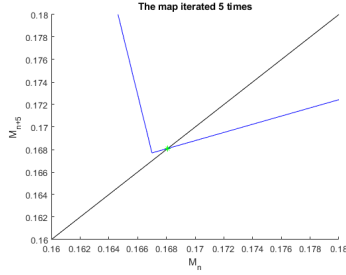


Figure 34: Close-ups of the 5-fold composition of the linear approximation with top point μ set to 0.167, 0.1677 and 0.1685. The range and domain of the close-up is $[0.16, 0.18]$. Green stars indicate stable fixed points and red dots indicate unstable fixed points.

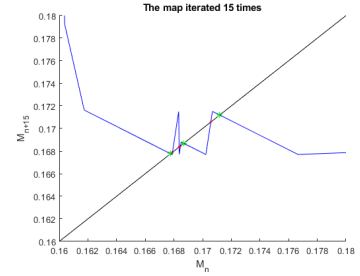
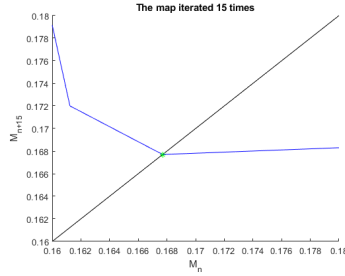
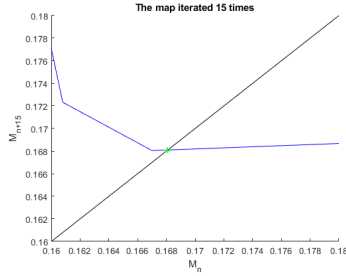


Figure 35: Close-ups of the 15-fold composition of the linear approximation with top point μ set to 0.167, 0.1677 and 0.1685. The range and domain of the close-up is $[0.16, 0.18]$. Green stars indicate stable fixed points and red dots indicate unstable fixed points.

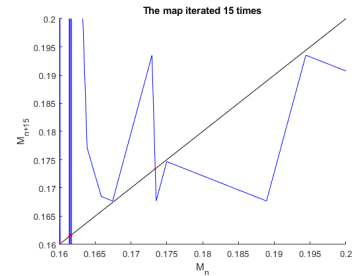
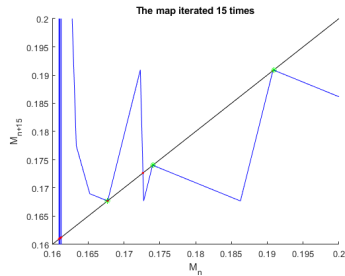
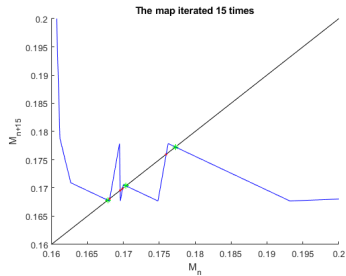


Figure 36: Close-ups of the 15-fold composition of the linear approximation with top point μ set to 0.17, 0.174 and 0.175. The range and domain of the close-up is $[0.16, 0.2]$. Green stars indicate stable fixed points and red dots indicate unstable fixed points.

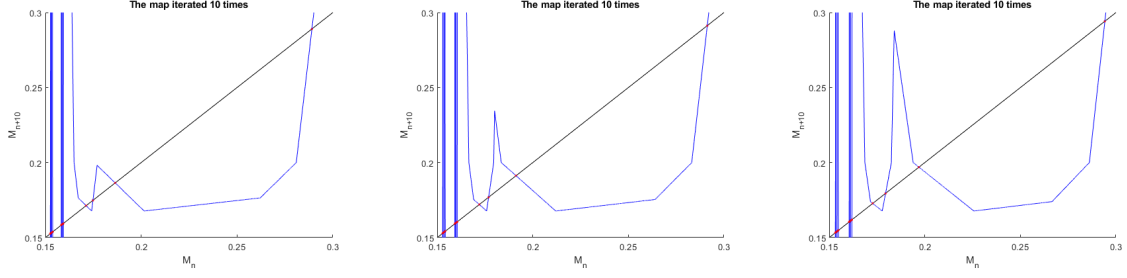


Figure 37: Close-ups of the 10-fold composition of the linear approximation with top point μ set to 0.177, 0.18 and 0.184. The range and domain of the close-up is $[0.15, 0.3]$. Green stars indicate stable fixed points and red dots indicate unstable fixed points.

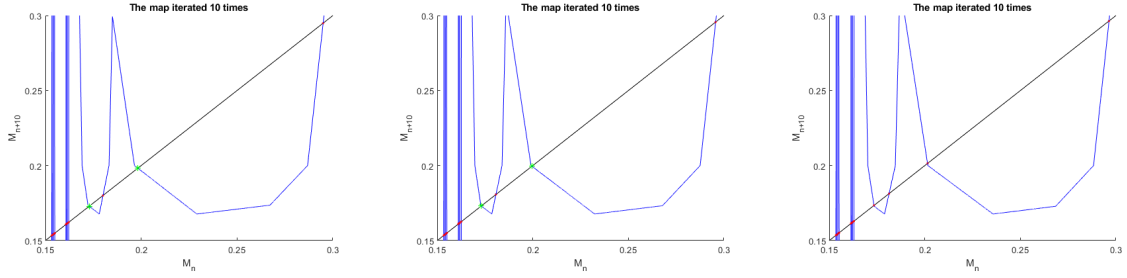


Figure 38: Close-ups of the 10-fold composition of the linear approximation with top point μ set to 0.185, 0.186 and 0.187. The range and domain of the close-up is $[0.15, 0.3]$. Green stars indicate stable fixed points and red dots indicate unstable fixed points.

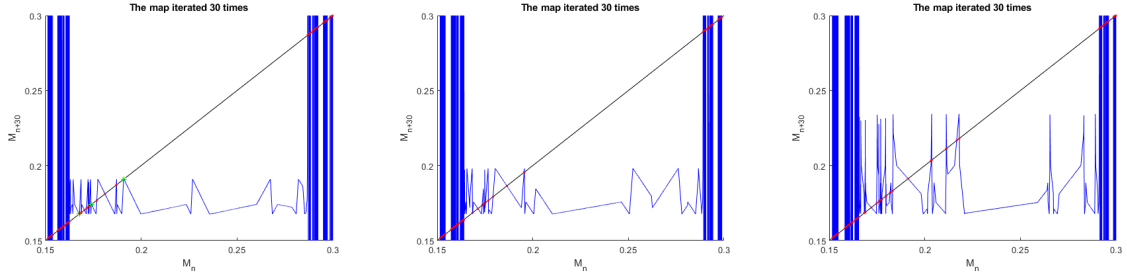


Figure 39: Close-ups of the 30-fold composition of the linear approximation with top point μ set to 0.174, 0.177 and 0.18. The range and domain of the close-up is $[0.15, 0.3]$. Green stars indicate stable fixed points and red dots indicate unstable fixed points.

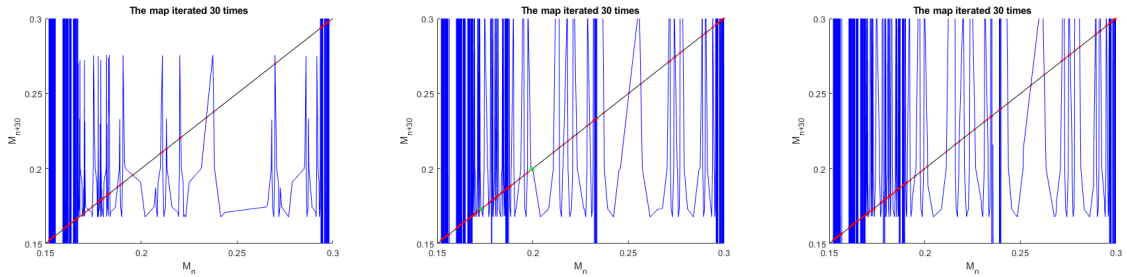


Figure 40: Close-ups of the 30-fold composition of the linear approximation with top point μ set to 0.183, 0.186 and 0.187. The range and domain of the close-up is $[0.15, 0.3]$. Green stars indicate stable fixed points and red dots indicate unstable fixed points.

6 Snap-back repellers

We continue with showing the existence snap-back repellers for linear approximations. Let us recall Definition 11 and follow the requirements as a recipe. We will start with the 7-piece linear approximations of section 3.2 and see if we can broaden towards the 7-piece linear approximation with shifting point of section 3.3.

6.1 Finding snap-back repellers for the linear approximation

We start with the initial 7-piece linear approximation f_7 with its only fixed point p . The first criterion is the definition states that p needs to be an expanding point in the neighbourhood $B_r(p)$. For this, f_7 needs to be differentiable inside $B_r(p)$. It is key to keep this neighbourhood large to have enough room to find a point $x_0 \in B_r(p)$ with p in its forward orbit.

Choosing the right neighbourhood such that p is an expanding point

We know from Section 3.2 that $p \approx 0.4496$ lies on the linear part between points given as (a_i, b_i) with $a_5 = 0.41$ and $a_6 = 0.46$. The piece-wise linear map f_7 only not differentiable on these points (a_i, b_i) . Therefore, the neighbourhood $B_r(p)$ needs to lie within the interval $(0.41, 0.46)$. We stay within this interval when we take $r = 0.01$.

For all values $x_0 \in B_{0.01}(p)$, we have a slope of $-0.62/0.05 = -12.4$. Since f_7 is a 1-dimensional map, the only eigenvalue of $Df_7(x)$ equals the slope. The absolute value of the slope easily exceeds 1 and thus we know that p is an expanding point of f_7 inside $B_{0.01}(p)$.

An approach with the backward orbits of p

To find a starting point of the orbit towards a snap-back repeller to fulfilling the second criterion, we need to find a point $x_0 \in B_{0.01}(p)$ not equal to p such that there exists an integer $k > 0$ with $f_7^k(x_0) = x_k = p$ and $Df_7^k(x_0) \neq 0$. To find this point, we will look into possible backward orbits of p . We will do this by choosing $x_0 = p$ and find $f_7^{-k}(x_0) = x_{-k} \in B_{0.01}(p)$ such that $x_0 \neq x_{-k}$, or better, $x_0 \neq x_{-1}$, to avoid equivalent orbits.

It is impossible to determine exactly the right point such that its forward orbit will hit p exactly. Therefore, we will investigate the backward orbit of p to search for a point within $B_{0.01}(p)$. The drawback of this method is that we need to consider the inverse of f_7 , which consists of two functions $f_{7,l}^{-1}$ and $f_{7,r}^{-1}$, described by equations (8) and (9). Therefore, the inverse can have two different answers for the same input. To denote the path of inverses we consider, we list these in the subscript. For instance $f_{i_1 \dots i_k}^{-k}(x)$, where $i_j \in \{l, r\}$ for all $j \in \{1, \dots, k\}$. An example of such a part of inverses is $f_r^{-1} \circ f_l^{-1}(x) = f_{lr}^{-2}(x)$.

Since precision is key, knowing that f_7 behaves chaotically, we determined values of possible backward orbits of p as fractions by keeping track of their integer numerator and denominators. Fortunately, this is relatively easy to do with linear approximations. For starters, we remark that the periodic point can be determined by using equation (6) for the fifth segment of the linear approximation. Namely, $f_7(p) = p = 0.94 + \frac{0.32-0.94}{0.46-0.41}(p - 0.41)$, $p \in [0.41, 0.46]$ results in $p = \frac{753}{1675}$. We also remark that $f_{7,r}^{-1}(p) = p$. We prefer to have that $f^{-1}(p) \neq p$ and thus we will start the backward orbit with a left inverse.

Our goal now is to find $x_{-k} f_{7,i_1 \dots i_k}^{-k}(p)$ such that $i_1 = l$ and $x_{-k} \in B_{0.01}(p)$. We separate the numerator and denominator to calculate x_{-k} with exact fractions using numerical trail and error. Hence we determine the backward orbit of every path of partial inverses and check if those fractions are within $B_{0.01}(p)$.

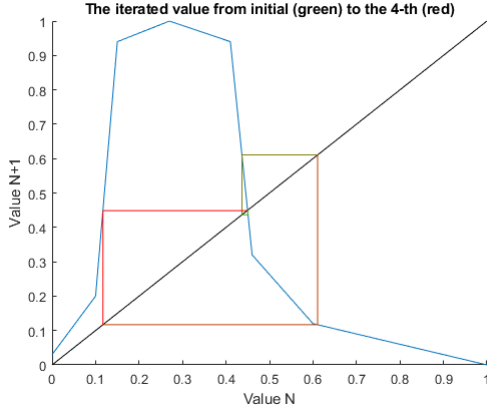


Figure 41: The graphical analysis of $f_{7,lr}^{-4}(p)$.

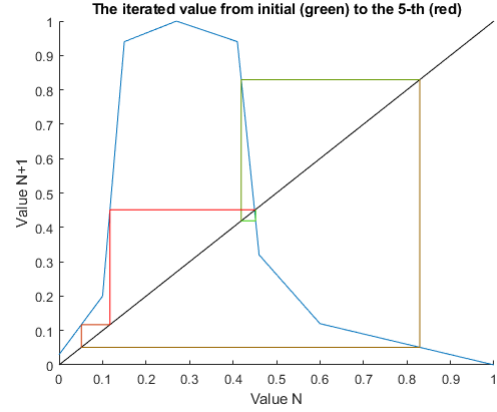


Figure 42: The graphical analysis of $f_{7,lr}^{-5}(p)$.

Numerically finding the orbits to prove that p is a snap-back repeller

The results of this approach gives us multiple answers. We will consider to two solutions for the smallest value k , namely $f_{7,lr}^{-4}(p) = \frac{322040977}{714695700} \approx 0.4506$ and $f_{7,lr}^{-5}(p) = \frac{5492019109}{12149826900} \approx 0.4520$. Figure 41 and 42 show that these points indeed map towards p after 4 and 5 iterations respectively. Their respective paths of partial inversions can also be derived by following their graphical analyses.

6.2 Proving chaos in the sense of Devaney with the snap-back repellers

With the previous result, however, we cannot yet say that the map f_7 is chaotic in the sense of Devaney. Recall that Theorem 1 requires that the five conditions of Definition 12 need to hold for our snap-back repeller.

Non-degeneracy of snap-back repellers

We first will show that p is a nondegenerate snap-back repeller by proving the last three items of Definition 11. Item 3 holds if we choose a neighbourhood such that $f(B_r(p)) \subset [0, 1]$. Since f_7 is strictly decreasing on the interval $(0.27, 1)$, this holds for all $r < p - 0.27$. Note that, by proving this item, we have shown indirectly that p is a regular expanding point, since p needs to be inside this open set.

Next, we note that the two starting points we found are interior points of $B_r(p)$. Therefore, there indeed exists a neighbourhood B_{δ_0} around $f_{7,lr}^{-4}$ and $f_{7,lr}^{-5}$ that is within the interior of $B_r(p)$, which proves the fourth item.

For the last item, we use a neighbourhood with radius $r = 0.00001$. We calculate the forward orbit of the boundaries of the neighbourhoods to see if p is indeed a interior point. From these calculations we indeed get the expected results, $p \in (f^4(f_{7,lr}^{-4} - 0.00001), f^4(f_{7,lr}^{-4} + 0.00001)) \approx (0.4427, 0.4564)$ and $p \in (f^5(f_{7,lr}^{-5} - 0.00001), f^5(f_{7,lr}^{-5} + 0.00001)) \approx (0.4379, 0.4612)$. Therefore, the last item is also proven and p is indeed a nondegenerate snap-back repeller.

6.3 Snap-back repeller in the linear approximations with shifting point

Now that we have proven that p is a nondegenerate snap-back repeller satisfying Definition 12, we proceed to Theorem 1. Note that the map f is only chaotic in the sense of Devaney on $\Lambda \subset U$, where U is any neighbourhood of p . Recall that f_7 had chaotic behaviour while $f_{7\mu}$ for all $\mu \leq 0.1677$ had stable period 5 solutions.

orbit path	uses top line-pieces	value	numerator	denominator
lrrr	no	0.45060	322040977	714695700
llrrr	no	0.45202	5492019109	12149826900
lrlrrr	no	0.45050	23825571943	52887481800
lllrrr	yes	0.45625	18239394383	39976849800
lrllrrr	no	0.45189	609427139849	1348630785900
llrlrrr	no	0.45043	404974094281	899087190600
lllrlrr	yes	0.44122	11758979597	26651233200
lrlllrrr	no	0.45204	9449252603597	20903777181450
lrlrlrrr	no	0.45050	3526222870049	7827347306400
lrlllrrr	yes	0.45389	2014101826513	4437430327800
llrllrrr	no	0.45064	37680616716263	83615108725800
llrlrrr	no	0.45169	10355930752433	22926723360300

Table 2: Starting points of the 12 orbits going towards p within 8 iterations of f_7 .

orbit path	value	numerator	denominator
lllrrr	0.45881	366836781727	799536996000
lllrlrrr	0.44504	14883585672803	33047529168000
lrlllrrr	0.45473	40356438316697	88748606556000

Table 3: Starting points of the 3 different orbits using top line-pieces of $f_{7\mu}$ with $\mu = 0.1677$.

We remark that the two orbits of f_7 leading to the snap-back repeller only had points in the orbit that use function lines that both f_7 and $f_{7\mu}$ have. Since Theorem 1 only claims chaos for a specific Cantor set, maybe we can find certain orbits towards p of f_7 that do not occur at $f_{7\mu}$ when $\mu = 0.1677$ for example, due to its stable behaviour.

12 orbits towards p for the map without and with shifting point

To answer this question, we first need to find more orbits, such that every line-piece of f_7 has a point of at least one orbit. To achieve this, we calculated all possible orbits towards p within 8 iterations. We did the same for $f_{7,\mu}$ with $\mu = 0.1677$.

It turns out that three of these orbits use the top line-pieces. Hence those orbits cannot be orbits of $f_{7\mu}$. By looking into the orbits of $f_{7\mu}$, the three orbits using its top line-pieces can be found and are shown in Table 3. Note that the middle orbit of that table needed an extra right inverse to get into the neighbourhood $B_{0.01}(p)$ compared to the seventh orbit in Table 2. The graphical analyses of both maps are shown in Figures 43 and 44, showing the use of the top line-pieces three times and their differences.

Due to slope near the fixed point p , right inverses of both f_7 and $f_{7\mu}$ converge to p rapidly. Thus every combination of left and right inverse at the start can eventually lead to points in any neighbourhood of p of radius $r > 0$. Also, every line-piece of both maps are used and, if we go far enough, any neighbourhood will have a point with a forward orbit towards p , even for the map $f_{7\mu}$.

Since every combination of inverses has a possibility to reach $B_r(p)$ for any $r > 0$, we can replicate the Cantor set for which both maps are chaotic. Like with the construction of the famous Cantor ternary set, we can choose between left or right inverses an infinite number of times, provided that we start with a left inverse. This confirms that there is indeed a Cantor set for which both maps f_7 and $f_{7\mu}$ act chaotically in the sense of Devaney.

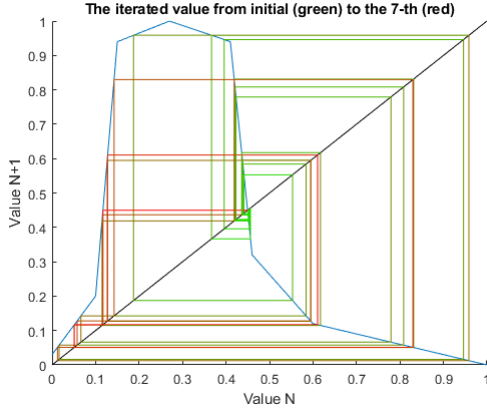


Figure 43: The graphical analyses of the points of Table 2 with map f_7 .

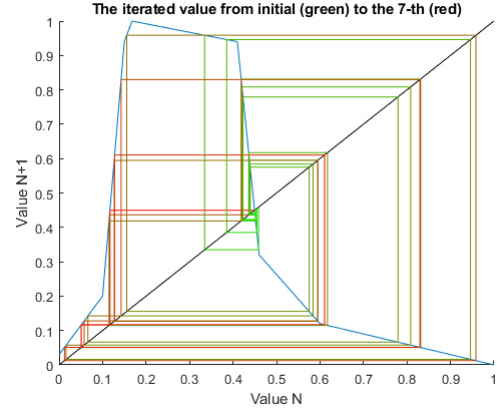


Figure 44: The graphical analyses of the points of Table 2 and 3 with map $f_{7\mu}$.

7 Predictability of extreme events

The toy model we are currently working with is specifically designed to approximate one variable of the ENSO model. Eastern equatorial Pacific temperatures play an important role in the El Niño phenomenon and therefore a toy model can be interesting. To determine the quality of its results, we need to be able to quantify them.

In the paper of Sterk et al. [SHR12], a methodology is presented to determine the predictability of extreme events of a dynamical system using preceding points with a certain time step. Determining the predictability of El Niño points using the toy model enables us to quantify the quality of the results of the toy model.

7.1 Applying the methodology on the toy model

We will start with applying the methods of the paper of Sterk et al. to the toy model. Before we start, we discuss some differences in approach between the approach used in the paper of Sterk et al. and the approach we will be taking.

Differences in approach

The approach we will be taking is fundamentally different. In the paper, predictability is determined by creating discrete models and we already consider a discrete model for which we want to determine predictability of extreme events.

The set of points denoting extreme events mentioned in the paper of Sterk et al. are chosen to be a certain percentile of the highest observed values. From those values, discrete time steps are used to determine a set of preceding values to determine predictability in that specific case. Both of these sets contain pieces of continuous trajectory.

With the toy model, we approach it the other way around. The toy model gives us the discrete steps towards extreme events. Therefore, the sets of extreme events in the case of the toy model are just points and not pieces of continuous trajectory.

The methodology used for the toy model

As is done in the paper of Sterk et al., we choose a sample of points S from the time series $O^+(x_0)$ of the map f . For this, we choose a sufficiently large sample size N .

$$S = \{f^k(x_0) \mid k \in \{1, \dots, N\}\}.$$

The goal is to determine the predictability of extreme events. First off, we need to determine what we consider as an extreme event. This will be the set E_q consisting of values of S that exceed a certain value q .

$$E_q = \{x \in S \mid x > q\}.$$

To examine the predictability of these extreme events, we need to determine the set of values $L_{\tau,q}$ that lead to an extreme event after a certain number of iterations. The number of iterations before an extreme event is called the lead time τ .

$$L_{\tau,q} = \{x \in S \mid f^\tau(x) \in E_q\}.$$

The predictability of extreme events is determined by the time- τ Lyapunov exponent of the lead time set L . The set of all these predictability values is denoted as $\Lambda_{\tau,q}$.

$$\Lambda_{\tau,q} = \{\lambda_\tau(x) \mid x \in L_{\tau,q}\}.$$

With the distribution of values in the set $\Lambda_{\tau,q}$ we will be able to classify the predictability of points $x \in S$ for some lead time τ , where points $x > q$ are considered extreme events. Negative values within $\Lambda_{\tau,q}$ indicate well-predictable extreme events, while positive are not well-predictable. The smaller the value, the better the predictability of that extreme event.

7.2 Insight in predictability calculations

The calculations for predictability in the case of the toy model are quite straightforward. To get a feeling of the calculations we will be doing numerically, we will treat some examples analytically.

Intervals of the lead time set

Specifically for El Niño maps, we can determine the intervals of the lead time sets $L_{\tau,q}$ with relative ease, especially when we consider El Niño points as extreme events. Therefore we will be discussing the set $L_{\tau,p}$. Recall from Section 4.2 that, for some El Niño map f , all points x such that $f^1(x) > p$ are within $(f_l^{-1}(p), p)$. Hence we know that $L_{1,p} = (f_l^{-1}(p), p)$.

Note that the borders of intervals of El Niño points are found with the left and right inverse of p and that $(f_l^{-1}(p), p) = (f_l^{-1}(p), f_r^{-1}(p))$. The same goes for higher values of τ . Therefore, we can obtain the intervals for lead time $\tau = 2$ by left and right inverting p one step further.

$$L_{2,p} = (f_{ll}^{-2}(p), f_{rl}^{-2}(p)) \cap (f_{rr}^{-2}(p), f_{lr}^{-2}(p)) = (f_{ll}^{-2}(p), f_l^{-1}(p)) \cap (p, f_{lr}^{-2}(p)).$$

Note, or check with any El Niño map, that with a left inverse, subsequent values become lower and with a right inverse, low values are followed by the highest values and high values become relatively low but are still larger than every left inverse outcome. This is important to note when we order the intervals. Continuing this process for any lead time τ will lead to a lead time set of

$$L_{\tau,p} = (f_{ll\dots l}^{-\tau}(p), f_{rl\dots l}^{-\tau}(p)) \cap \dots \cap (f_{rl\dots lr}^{-\tau}(p), f_{ll\dots lr}^{-\tau}(p)).$$

General term for predictability of El Niño points

With analytic calculations, we can do not need to consider a sample size or a specific orbit. Hence, for any lead time τ , we can represent the set of predictability values of El Niño points as

$$\Lambda_{\tau,p} = \left\{ \frac{1}{\tau} \log(f^{\tau'}(x)) \mid (f_{ll\dots l}^{-\tau}(p), f_{rl\dots l}^{-\tau}(p)) \cap \dots \cap (f_{rl\dots lr}^{-\tau}(p), f_{ll\dots lr}^{-\tau}(p)) \right\}.$$

7.3 Predictability of extreme events of linear approximations

The two El Niño maps we are going to use for numerical computation is the 7-piece linear approximation f_7 and the 50-piece linear approximation f_{50} . We will use a sample size of $S = 10,000$ and consider lead times $\tau = 1$ until $\tau = 6$.

For selection of events for this experiment, we use the whole set S , the set of El Niño points and the events such that values exceed 0.95 and 0.99. Therefore, we will have $q = 0$ and $q = p$ in the first two cases and $q = 0.95$ and $q = 0.99$ in the last two cases. The fixed point p of f_7 equals $\frac{753}{1675} \approx 0.4496$ and p of f_{50} equals $\frac{8457}{19000} \approx 0.4451$. We remark that for f_7 , 26.5 percent are El Niño points and for f_{50} only 19.2 percent. For the case $q = 0.95$ more than half of the highest El Niño points are considered and for $q = 0.99$ it is the highest twelfth for f_7 and the highest sixth for f_{50} .

Result of these experiments are shown in Figure 45 to 52. With these boxplots, the median is indicated with the middle red line, the box shows the range of the middle half values and the whiskers denote the minimum and maximum value excluding outliers, which are shown with red pluses.

Results of the individual experiments of the map f_7

The results for predictability of the sample set considering the map f_7 , given in Figure 45, show a relative steady median which stays between 0 and 0.5. Furthermore, it shows a wide range of Lyapunov exponent values that becomes smaller when τ becomes larger. This is because different parts of the map f_7 map the initial point and the Lyapunov exponent averages the slopes of these parts. For this case, the events seem to be relatively good predictable when $\tau = 3$ opposed to other values for τ , although the differences are small.

Secondly, we have the results for predictability of El Niño points, shown in Figure 46. The behaviour of the range does not differ much from the case considering all values, but the box indicating the middle half values differs a lot. Instead of considering all points, we now only consider El Niño points as extreme events, which is 26.5 percent of the points in our sample set S . The median jumps from -0.5 to 1 and goes back to under 0.5, indicating that there are many well-predictable El Niño points for $\tau = 1$ and many with a bad predictability when τ equals 2, 3 or 4.

In the last two cases, the set of points such that $x > 0.95$ and $x > 0.99$ are considered as extreme events. These are shown in Figure 47 and 48 respectively. Due to the smaller and more specific selection of points combined with the basic shape of the map f_7 , we start with a small range of predictability values, which gets larger as τ progresses. In fact, for $\tau = 1$, only two distinct values occur in these case, since these points only come from the two highest lines of the map f_7 . When $\tau = 2$, the values are still quite well-predictable and after some relatively bad predictable values after that, the Lyapunov dives under 0.5 again.

Predictability results of f_{50} compared to f_7

Figure 49 to 52 show the result for the predictability of values of the map f_{50} . The maps f_7 and f_{50} more or less show the same behaviour. The differences between these maps are explainable from the characteristics of these maps.

The medians of predictability values of the map f_7 act somewhat jumpy at some points. Due to the more smooth shape of the map f_{50} , these medians are less fidget. This is well-visible for the predictability of the whole sample set. For f_{50} , the median has a far more constant behaviour opposed to that of the map f_7 . This behaviour is also visible in the cases of extreme events. It is not hard to imagine a smooth curve through these medians.

Some behaviour is emphasized with this more smooth approximation. In general, the more selective we are with extreme events, the better their predictability. For the case when $q = 0.99$ and $\tau = 1$, these values average around -2 . Note that the lead time for which values are least well-predictable gets larger as the selection of extreme events becomes more strict. Thus, higher values are better predictable and, if the selection of the set of extreme events is really strict, these stay well-predictable for higher values of τ .

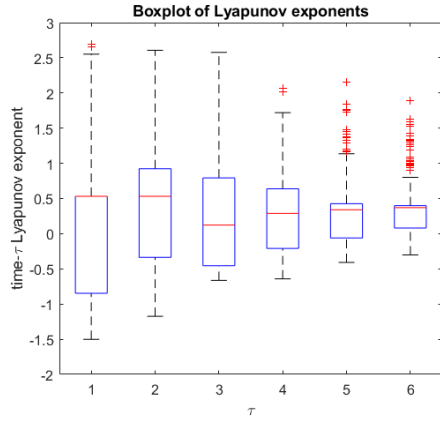


Figure 45: A boxplot of predictability values of the sample set with τ from 1 to 6 iterations for the map f_7 .

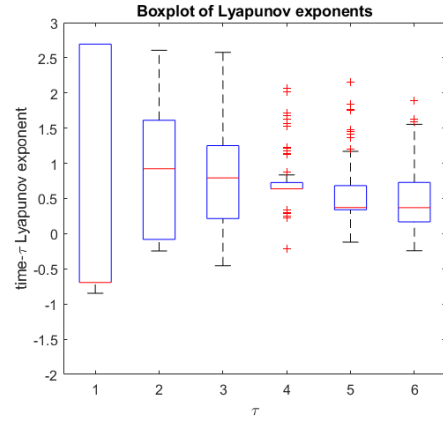


Figure 46: A boxplot of predictability values of El Niño points with τ from 1 to 6 iterations for the map f_7 .

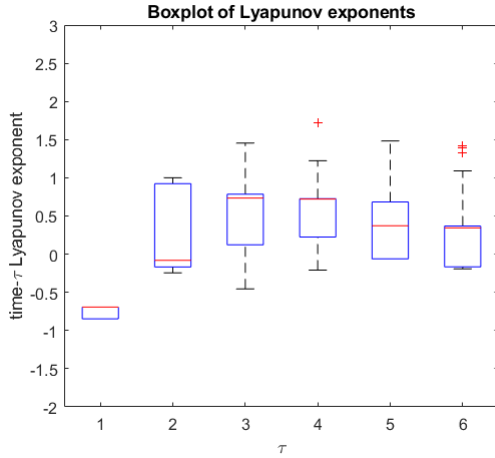


Figure 47: A boxplot of predictability values of points $x > 0.95$ with τ from 1 to 6 iterations for the map f_7 .

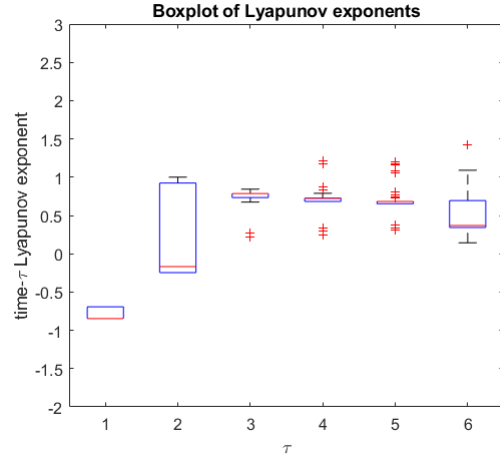


Figure 48: A boxplot of predictability values of points $x > 0.99$ with τ from 1 to 6 iterations for the map f_7 .

Concluding predictability of extreme events

Both f_7 and f_{50} show about the same behaviour. Because the latter map is more smooth, its predictability values are more smooth. A nice feature is that their medians tend to their Lyapunov exponent as τ progresses.

To conclude, we have seen that with more strict sets of extreme events more well-predictable points occur and values are well-predictable for larger values for τ . For both linear approximations, when $q > 0.95$, values are mostly well-predictable for lead time $\tau = 1$ and when $q > 0.99$ for the map f_{50} , values are also mostly well-predictable for lead time $\tau = 2$. With mostly well-predictable, we mean that more than 75 percent of the values have a negative time- τ Lyapunov exponent. In the case $q = 0.973$, that is more than 30 percent of El Niño points, all extreme events have a negative predictability value for $\tau = 1$ and hence are all well-predictable.

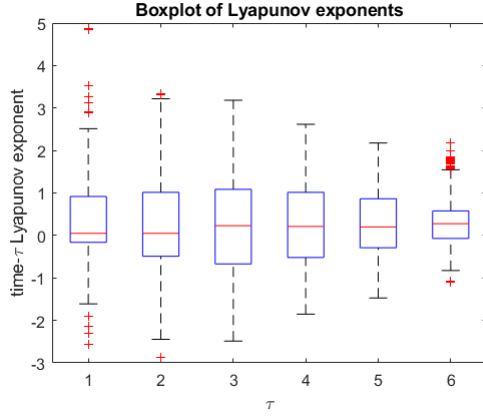


Figure 49: A boxplot of predictability values of the sample set with τ from 1 to 6 iterations for the map f_{50} .

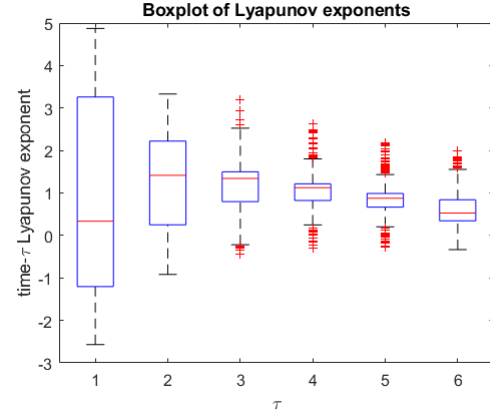


Figure 50: A boxplot of predictability values of El Niño points with τ from 1 to 6 iterations for the map f_{50} .

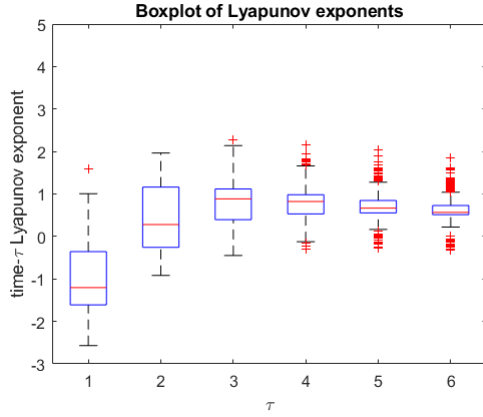


Figure 51: A boxplot of predictability values of points $x > 0.95$ with τ from 1 to 6 iterations for the map f_{50} .

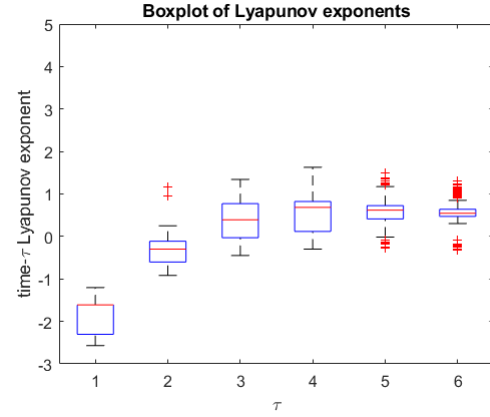


Figure 52: A boxplot of predictability values of points $x > 0.99$ with τ from 1 to 6 iterations for the map f_{50} .

8 Discussion and conclusion

After investigation several properties and characteristics of the toy model for the ENSO model, we will conclude by answering the main question after we discuss the four sub-questions.

Can it be useful to employ a toy model over the ENSO model?

- *Can we simplify the ENSO model into a toy model that is able to simulate one variable accurately?*
- *How can we determine the difference in quality of results between the toy model and the ENSO model?*
- *What are the possible benefits of using such a toy model instead of the ENSO model?*
- *Are there possible downsides on using a toy model instead of the ENSO model?*

Quality of the results

With the toy model, described in Section 3, we are indeed able to make a model using local maxima of the eastern equatorial Pacific sea surface temperatures T_2 of the ENSO model. The accurate example of the 50-piece linear approximation has El Niño periods 2 to 7, which corresponds to El Niño. More relevant, the ENSO model has an average time distance between El Niño phenomena of 4 to 5 years, which is similar to this example. Therefore it is possible to simplify the ENSO model with a toy model that models one variable such that dynamics like the El Niño period are similar.

In Section 7, we were able to quantify predictability of map approximations. From the results of the predictability of the toy model we deduced that extreme cases of El Niño points were very well-predictable. The results of that experiment say that the top half temperatures T_2 of El Niño phenomena are 75 percent well-predictable and the highest 30 percentile are all well-predictable with a lead time of one iteration. In practice this would mean that extreme cases of the El Niño phenomenon can be well predicted a year in advance with a toy model. To determine the quality between the toy model and the ENSO model, such a predictability analyses must be preformed on the ENSO model.

Overall, the toy model can deliver accurate and qualitative good results by using only one variable of the ENSO model, answering the first two sub-questions. To give a complete answer on the second sub-question, the ENSO model also needs to be analyzed.

Practical benefits and downsides

With the ENSO model, a wide scope of variables and their dependence on each other is considered to model the El Niño-Southern Oscillation as well as possible. But with using a toy model, we are able to focus on a specific variable. This simplification of the ENSO model saves time, calculation capacity and decreases the need of comprehensive knowledge on the El Niño phenomenon and dynamical systems, making it easier to understand and more accessible to work with.

Using a simplification, however, causes that not every aspect is taken into account. Despite that variable values are probably in line with the ENSO model, it is quite possible that, for example, order or frequency of variable values deviate far from reality because the underlying process is not taken into account. A close look into differences of trajectories of the toy model and the ENSO model can disclose this matter.

Therefore, the possibility of using a toy model can provide an elementary and straightforward way of modelling, which can be useful in many situations where time, capacity and knowledge are not abundant.

Research in context of comparable work

Before we conclude the thesis, we shortly view the methods we have treated in the light of other methods for simplifying models. Most of these articles concentrate on a generic way to simplify a

model, although the goal is the same: maximizing the utility of the model by keeping both fidelity and simplicity as high as possible [E07]. A challenging task since these are conflicting characteristics.

Most of these generic simplifications concentrate on the more idle parts of a model. An example is the inactive junction concept, often used in articles on mechanical engineering [E07] & [EFS06]. More mathematical methods to achieve a simpler model is presented in the book [W90] by reducing dimensionality and eliminating non-linearity. The approach of this thesis amounts to the same thing. An example of linearization through a Poincaré section is found in [PY01]. The purpose of that paper is to find an alternative with the use of a formal change of variables such that the transformed differential equation defines a system with a new symmetry.

In this context, we see that roughly the same methods are used to achieve the goal of simplifying a model. The toy model, however, approaches this more ruthlessly by concentrating on one variable and ignoring the others. Therefore, the fidelity of the model is at stake. This is greatly compensated with the degree of simplicity, but we do not know if it is worth it compared to other methods.

Conclusion and further research possibilities

With the answers on the four sub-questions, we conclude that employing a toy model over the ENSO model indeed can be useful.

There are multiple areas which have further research possibilities. Examples to this are comparing different toy model approaches, comparing toy models on distinct dynamical systems, calculating the time benefits of using a toy model on a certain dynamical system and calculating the accuracy loss due to a toy model. Other dynamical systems to apply and compare with this approach can include the Lorenz attractor among others. A couple of examples can be found in [SHR12].

Applying existing methods, shortly discussed in the previous section, on the ENSO model and comparing them to the toy model can give more sight on how well this simplification preforms compared to these other methods. Measuring the fidelity/simplicity trade-off can further determine the exact usefulness of the toy model of the ENSO model.

References

- [BT09] Broer, H., Takens, F., *Dynamical systems and chaos*, Epsilon Uitgaven, 2009.
- [D89] Devaney, R. L., *An introduction to chaos dynamical systems*, Addison-Wesley Publishing Company inc., 2nd edition, 1989.
- [E07] Ersal, T., *Realization-Preserving Simplification and Reduction of Dynamic System Models at the Graph Level*, University of Michigan, 2007.
- [EFS06] Ersal, T., Fathy, H. K., Stein, J. L., *Structural Simplification of Modular Bond-Graph Models Based on Junction Inactivity*, American Society of Mechanical Engineers, Dynamic Systems and Control Division, IMECE2006, 1117-1128, 2006.
- [HSD04] Hirsch, M. W., Smale, S., Devaney, R. L., *Differential equations, dynamical systems and an introduction to chaos*, Elsevier Academic Press, 2nd edition, 2004.
- [M78] Marotto, F. R., *Snap-back repellers imply chaos in \mathbf{R}^n* , Journal of Mathematical Analysis and Applications, 63, 199–223, 1978.
- [M05] Marotto, F. R., *On redefining a snap-back repeller*, Chaos, Solitons and Fractals, 25, 25-28, 2005.
- [PY01] Palacián, J., Yanguas, P., *Analytical approach for simplifying dynamical systems of polynomial type*, Mathematics and Computers in Simulation, 57, 291–305, 2001.
- [R99] Robinson, C., *Dynamical systems: stability, symbolic dynamics, and chaos*, Taylor & Francis Group, 1999.
- [SC04] Shi, Y. and Chen, G., *Chaos of discrete dynamical systems incomplete metric spaces*, Chaos, Solitons and Fractals, 22, 555-571, 2004.
- [SC05] Shi, Y. and Chen, G., *Discrete chaos in Banach spaces*, Science in China series A mathematics (English version), 48, 222-238, 2005.
- [SHR12] Sterk, A. E., Holland, M. P., Rabassa, P., Broer, H. W. and Vitolo, R., *Predictability of extreme values in geophysical models*, Nonlinear Processes in Geophysics, 19, 529–539, 2012.
- [SY05] Shi, Y. and Yu, P., *Chaos induced by snap-back repellers and its applications to anti-control of chaos*, Proceedings of the 4th DCDIS International Conference on Engineering Applications and Computational Algorithms, 364–369, 2005.
- [S94] Strogatz, S. H., *Nonlinear dynamics and chaos*, Perseus Books Publishing, 1994.
- [TJ02] Timmermann, A. and Jin, F.-F., *A Nonlinear Mechanism for Decadal El Niño Amplitude Changes*, Geophysical Research Letters, 29(1), 3-1 - 3-4, 2002.
- [W90] Wiggins, S., *Introduction to Applied Nonlinear Dynamical Systems and Chaos*, Springer-Verlag, 1990.
- [WMO14] World Meteorological Organisation, *El Niño/Southern Oscillation*, World Meteorological Organization, 1145, 2014.

Appendix

Proof of Corollary 1

Proof.

1. The map f is onto, hence $\exists x \in [0, 1]$ such that $f(x) = 1$. Consider the point t from requirement 4 of Definition 14. $\forall \epsilon_1, \epsilon_2$ such that $t > \epsilon_1 > \epsilon_2 > 0$, we have $f(t - \epsilon_1) < f(t - \epsilon_2)$ and $f(t + \epsilon_2) > f(t + \epsilon_1)$. Due to continuity, we must have $f(t - \epsilon_2) < f(t)$ and $f(t) > f(t + \epsilon_2)$. If $f(t) < 1$, then f cannot be onto. Hence $f(t) = 1$.

Next consider the only fixed point p of f and the continuous function $g(x) = f(x) - x$. Assume $t > p$, then $\forall x > t$ we have $f(x) < 1$. Now $\forall t < 1$, $g(t) = 1 - t > 0$ and $g(1) = f(1) - 1 < 1 - 1 = 0$.

The **intermediate value theorem**, which can be found in [D89, p.10], state that for any continuous function $h : [a, b]$ with $h(a) = u$ and $h(b) = v$ and for any w between u and v . There exists a c between a and b such that $h(c) = w$.

By the intermediate value theorem, because $g(t) > 0$ and $g(1) < 0$, $\exists p^* \in (t, 1)$ such that $g(p^*) = 0$. This implies that $f(p^*) = p^*$, which means that it is another fixed point of f . This is not possible and thus our assumption $t > p$ is not true.

Now consider the point $q \neq p$ with $f(q) = p$ from requirement 3 of Definition 14 and assume that $t < q$. Then, because $t < q < p$ and both q and p must be in the decreasing part of f , we must have $f(q) > f(p)$. But $f(q) = f(p) = p$. Therefore we need to have that $t \geq q$.

Next, assume that $t = p$. This implies $p = f(p) = f(t) = 1$. But since $p \in (0, 1)$, that is also not possible.

Lastly assume that $t = q$. This means that $f(t) = f(q) = p$, implying again that $p = f(t) = 1$, which leads to a contradiction.

If we subtract all impossible points and intervals from $[0, 1]$, we see that we must have that $t \in (q, p)$. \square

2. Recall that f is increasing $\forall x \in (0, t)$ and decreasing $\forall x \in (t, 1)$. But since f is onto, we must have that $\exists x \in [0, 1]$ such that $f(x) = 0$.

Due to the increasing nature of f at $(0, t)$, we have that $\forall x > 0$, we must have that $0 < f(x)$. This is because $\forall x^*$ such that $0 < x^* < x$ we have $f(x^*) < f(x)$. Hence the only option for all $x \in [0, t)$ to have $f(x) = 0$ is $x = 0$. This would mean that $f(0) = 0$ and hence it would be a fixed point. This yields a contradiction.

We consider f at the interval $(t, 1)$, which is decreasing. Here $\forall x < 1$, we have that $0 < f(x)$, because $\forall x^*$ such that $1 > x^* > x$ we have $f(x^*) < f(x)$. Therefore, only $x = 1$ can be an option to be such that $f(x) = 0$. Since this is the only option and considering that f is onto, we have that indeed $f(1) = 0$ for every El Niño map. \square

3. The only fixed point is $p \in (0, 1)$ and we have proven that the point such that $f(t) = 1$ has $t \in (q, p)$. We again consider the function $g(x) = f(x) - x$. $\forall x < p$ we have that $g(x) \neq 0$ because p is the only fixed point. By the intermediate value theorem, we know that $\forall x < p$ either $g(x) < 0$ or $g(x) > 0$. Otherwise there needed to be a $x < p$ for which $g(x) = 0$, indicating another fixed point. We already have proven that $f(t) = 1$, thus $g(t) = 1 - t > 0$. Therefore $g(x < p) > 0$ and hence $x < f(x < p)$.(★)

We also know that $f(x)$ is decreasing $\forall x \leq q < t$. Hence $\forall x, x^* \in [0, q]$, we have $x < x^*$ implies $f(x) < f(x^*)$. If we take $x^* = q$, we get $x < q$ implies $f(x) < f(q) = p$.

By combining the previous two arguments, we get that if $x < q$, then $x < f(x) < p$. \square

4. By argument (\star) , we already know that $\forall x \in (q, p)$, we have $x < f(x)$. We also know that $\forall x, x_1 \in [q, t)$, we have $x_1 < x$ implies $f(x) > f(x_1)$ and $\forall x, x_2 \in (t, p]$, we have $x < x_2$ implies $f(x) > f(x_2)$. By taking $x_1 = q$ and $x_2 = p$, these arguments imply that both $x \in (q, t)$ and $x \in (t, p)$ imply $f(x) > p$. If we also note that $f(t) = 1 > p$, we have proven that $\forall x \in (q, p)$, we have $f(x) > p$. \square
5. Since $t < p$, we know that $f(x \geq p)$ is decreasing. Hence $\forall x, x^* \in [p, 1]$, we have $x > x^*$ implies $f(x) < f(x^*)$. If we take $x^* = p$, we get $x > p$ implies $f(x) < f(p) = p$. \square
6. (\Rightarrow) , x is an El Niño point $\Leftrightarrow f(x) < x$. If $x < p$, then $x < f(x)$ by item 3 and 4 of Corollary 1. If $x = p$, then $x = f(x)$. Hence the only option left is $p < x$. By item 5 of Corollary 1, this is indeed true.
 (\Leftarrow) , $p < x$ implies $f(x) < p$ by item 5 of Corollary 1. From this, $f(x) < x$ instantly follows, which implies that x is indeed an El Niño point. \square
7. This statement is equivalent to proving that there does not exist an El Niño point with period 1, that is, there does not exist El Niño points x_1, x_2 such that $f(x_1) = x_2$. We recall that x is an El Niño point if and only if $p < x$.
 Assume x_1 is an El Niño point $\Rightarrow p < x_1 \Rightarrow f(x_1) = x_2 < p \Rightarrow x_2$ cannot be an El Niño point. Thus the El Niño point after x_1 will have a period of at least 2.

Matlab code

TMvalues.m

```
1
2 function [] = TMvalues(Map, Initial_value , Transient , Max_iter ,
   Plot_until_iter , Number_of_segments , Combined_figure)
3
4 % Toy Model values
5 %
6 % This function generates iterated values of a map and plots its forward
7 % orbit and histograms of values.
8 %
9 % Map          Enter a function handle to evaluate.
10 %
11 % Initial_value Enter an initial value to evaluate over the map.
12 %
13 % Transient     The number of iterations that will be omitted from
14 %               the results for the values and El Nino periods. When
15 %               this is set to -1, the initial value will be added.
16 %
17 % Max_iter      The maximum number of iterations for which the
18 %               values will be generated.
19 %
20 % Plot_until_iter Set the number of iterations for which the iteration
21 %               lines , lines that visualize the iteration process ,
22 %               need to be plotted.
23 %
24 % Number_of_segments The number of segments the unit interval will be
25 %               divided for the purpose of displaying the
26 %               distribution of all generated values.
27 %
28 % Combined_figure Set this value to 1 to let all 4 plots be in the
29 %               same figure. Otherwise, all plots will have their
30 %               own figure.
31
32 clf
33 close all
34
35 mapPlot = figure;
36
37 if Combined_figure == 1
38     histPlot = figure;
39 else
40     valHist = figure;
41     valScat = figure;
42     distHist = figure;
43     distScat = figure;
44 end
45
46 % Check if Initial_value , Max_iter , Plot_until_iter and Map_iter
47 % have valid values.
```

```

48 |
49 | if (Initial_value < 0) || (Initial_value > 1)
50 |     disp('! Not an input within the closed unit interval into variable "
      Max_iter" of function "TM".')
51 | end
52 |
53 | if (Max_iter < 0) || (Max_iter ~= round(Max_iter))
54 |     disp('! No non-negative integer input into variable "Max_iter" of
      function "TM".')
55 | end
56 |
57 | if (Plot_until_iter < 0) || (Plot_until_iter ~= round(Plot_until_iter))
58 |     disp('! No non-negative integer input into variable "Plot_until_iter"
      of function "TM".')
59 | end
60 |
61 | if Max_iter < Plot_until_iter
62 |     disp('! "Plot_until_iter" has a higher value than "Max_iter" and will
      be set equal.')
63 |     Plot_until_iter = Max_iter;
64 | end
65 |
66 | % Check whether the variable is a function handle and plot the function.
67 |
68 | figure(mapPlot);
69 |
70 | hold on
71 |
72 | if isa(Map, 'function_handle')
73 |     map = Map;
74 |     x = 0: .001: 1;
75 |     y = zeros(1, 1001);
76 |     for it =1: 1001
77 |         y(it)= map(x(it));
78 |     end
79 |
80 |     plot(x,y)
81 |
82 | else
83 |     disp('! No function handle input into variable "Map" of function "TM
      ".')
84 |
85 | end
86 |
87 | solArray = zeros(Max_iter - Transient , 1); % Generate large enough array.
88 | iterValue = Initial_value; % Let initial value be the first value.
89 |
90 | % When the transient is choosen to be -1, the initial value will be added
91 | % to the solution array.
92 |
93 | if -1 == Transient
94 |

```

```

95     solArray(1, 1) = iterValue;    % For the case the initial
96 end
97
98
99 plot([0, 1], [0, 1], "k");        % Plot the identity line (black).
100
101 highCount = -100;                % Another result will be the distance between high
102 highCountArray = [];            % values. To make sure it starts counting from a
103                                % El Nino point, the counter starts with -100.
104                                % Later, checks will be made if it is positive.
105
106 % Fill the solution array, plot the iteration lines (green to red) and
107 % fill the El Nino period array.
108
109 for iter = 1: Plot_until_iter
110
111     iterValueNew = map(iterValue);
112
113     if iter > Transient
114
115         solArray(iter - Transient, 1) = iterValue; % Filling solArray.
116     end
117
118     plot([iterValue, iterValue],...    % The line from the map
119          [iterValue, iterValueNew],...  % to the identity map.
120          'color', [iter/Plot_until_iter 1-iter/Plot_until_iter 0])
121
122     plot([iterValue, iterValueNew],...  % The line from the id
123          [iterValueNew, iterValueNew],... % map to the next point.
124          'color', [iter/Plot_until_iter 1-iter/Plot_until_iter 0])
125
126 % A El Nino point is reached if the next value is lower.
127
128     if iterValue > iterValueNew
129
130         % First we make sure that we start with an El Nino point.
131
132         if highCount > 0
133             highCountArray = [highCountArray, highCount];
134         end
135
136         highCount = 1; % If we would have two consecutive El Nino
137                        % points (not possible), then the distance is one.
138     else
139         highCount = highCount + 1; % When the current value isn't an El
140                                    % Nino point, the distance increases.
141     end
142
143     iterValue = iterValueNew;
144 end
145
146 title(['The iterated value from initial (green) to the ', ...

```

```

147     num2str(Plot_until_iter), '-th (red)']])
148 xlabel('Value N')
149 ylabel('Value N+1')
150
151 hold off
152
153 % Fill-up the remaining space in the array that doesn't need plotting.
154
155 for iter = Plot_until_iter + 1: Max_iter
156
157     iterValueNew = map(iterValue);
158
159     if iter > Transient
160
161         solArray(iter - Transient, 1) = iterValue; % Filling solArray.
162     end
163
164     if iterValue > iterValueNew
165
166         if highCount > 0
167             highCountArray = [highCountArray, highCount];
168         end
169         highCount = 1;
170
171     else
172         highCount = highCount + 1;
173
174     end
175
176     iterValue = iterValueNew;
177 end
178
179 % Creating a histogram and a scatterplot of the all iterated values and
180 % El Nino periods.
181
182 if Combined_figure == 1
183     figure(histPlot);
184     tiledlayout(2, 2)
185     nexttile
186 else
187     figure(valHist);
188 end
189
190 histogram(solArray, Number_of_segments)
191 title('The occurance of values')
192 xlabel('value')
193 ylabel('Number of occurrences')
194
195 if Combined_figure == 1
196     nexttile
197 else
198     figure(valScat);

```

```

199 end
200
201 histogram(highCountArray)
202 title('The occurrences of El Nino periods')
203 xlabel('El Nino period')
204 ylabel('Number of occurrences')
205
206 if Combined_figure == 1
207     nexttile
208 else
209     figure(distHist);
210 end
211
212 if Max_iter > 100
213     scatter(1 + Transient: length(solArray) + Transient, solArray, "b.")
214 else
215     plot(1 + Transient: length(solArray) + Transient, solArray, "b.-")
216 end
217
218 title('Chronology of iterated values')
219 xlabel('n')
220 ylabel('M_n')
221
222 if Combined_figure == 1
223     nexttile
224 else
225     figure(distScat);
226 end
227
228 if Max_iter > 100
229     scatter(1: length(highCountArray), highCountArray, "b.")
230 else
231     stem(1: length(highCountArray), highCountArray, "b.")
232 end
233
234 title('Chronology of El Nino periods')
235 xlabel('El Nino points')
236 ylabel('El Nino periods')
237 ylim([min(highCountArray)-1 max(highCountArray)+1])

```

TMcomposition.m

```

1 function [Periodic_points, Slope] = TMcomposition(Map,...
2     Composition_fold, Precision, Figures, Keep_figures,...
3     Min_x, Max_x, Min_y, Max_y)
4
5 % Toy Model composition
6 %
7 % This function determines the periodic points and their slopes of a
8 % certain composition of the given map.
9 %

```

```

10 % Map                Enter a function handle to evaluate.
11 %
12 % Composition_fold    If this number is non-zero, it will find the
13 %                    periodic points of this number of iterations for the
14 %                    choosen map.
15 %
16 % Precision           The precision used for calculating periodic points.
17 %                    Use a decimal number to determine the step size.
18 %
19 % Figures             If non-zero, a plot is shown of the requested
20 %                    composition of the map, showing stable and unstable
21 %                    periodic points.
22 %
23 % Keep_figures        If zero, it removes previously made figures.
24 %
25 % Min_x,... , Max_y   The range used for the figure.
26
27
28 if nargin <= 3
29     Figures = 1337;
30 end
31
32 if nargin <= 4
33     Keep_figures = 1337;
34 end
35
36 if nargin <= 5
37     Min_x = 0;
38     Max_x = 1;
39     Min_y = 0;
40     Max_y = 1;
41 end
42
43 if Keep_figures < 1
44     clf
45     close all
46 end
47
48 map = Map;
49
50 if Figures > 0
51     iterPlot = figure;
52
53     if Figures < 2
54         mapPlot = figure;
55
56 % Choosing a map and plotting it.
57
58         figure(mapPlot);
59
60 hold on
61

```

```

62 if isa(Map, 'function_handle')
63     x = 0: .001: 1;                                % This map is not directly plotted like
64     y = zeros(1, 1001);                             % the previous two. The y coordinates
65     for it =1: 1001                                  % are generated using its map function.
66         y(it)= map(x(it));
67     end
68
69     if Figures < 2
70         plot(x,y)
71     end
72 else
73     disp('! No function handle input into variable "Map" of function "TM
74         "."')
75 end
76
77 hold off
78
79 end
80
81 % Plot the n-th iteration of the map and find its periodic points.
82
83 figure(iterPlot);
84 hold on
85 plot([0, 1], [0, 1], "k")                          % Plot the identity map in black.
86
87
88     xlim([Min_x Max_x])
89     ylim([Min_y Max_y])
90
91 fprintf('The %3.0f times iterated map is checked for periodic points.\n'
92     ,...
93     Composition_fold)
94 end
95
96 Periodic_points = NaN(2^Composition_fold - 1, 1);
97 Slope = Periodic_points;
98 perPointCount = 0;
99
100 for step = Min_x: Precision: Max_x - Precision
101     iterStep1 = step;                                % Create variable for the
102     iterStep2 = step + Precision;                    % begining and end of each
103                                                         % interval to iterate with.
104
105     % The iteration of both values.
106     % Min and max statements are needed to strictly stay within unit
107     % interval to avoid machine accuracy problems.
108
109     for mapIter = 1: Composition_fold
110         iterStep1 = min(1, max(0, map(iterStep1)));
111         iterStep2 = min(1, max(0, map(iterStep2)));
112     end

```

```

112
113 % Plot the current interval of the map in blue.
114
115 if Figures > 0
116     plot([step, step+Precision], [iterStep1, iterStep2], "b")
117 end
118
119 % When an interval contains an odd number of periodic points (we
120 % expect the uses to choose an appropriate precision to avoid the
121 % possibility of more than one periodic points per interval), the
122 % difference between current and next step changes sign. The product
123 % of this difference at the start and the end of the interval will
124 % indicate an odd number of periodic points when it is non-positive.
125
126
127 if (iterStep1 - step) * (iterStep2 - (step + Precision)) <= 0
128
129     % The stability of a periodic point depends on the slope. If the
130     % absolute values of the slope of the interval is smaller than 1,
131     % then it will be stable (if the precision is appropriate).
132
133     slope = (iterStep2 - iterStep1) / Precision;
134
135     % We can locate the periodic point with higher precision if we
136     % consider that at that point, we must have step = iterStep, thus
137     % perPoint = step + x = iterStep1 + x * slope
138     %      x * (1 - slope) = iterStep1 - step
139     %      x = (iterStep1 - step) / (1 - slope)
140     % Hence,      perPoint = step + (iterStep1 - step) / (1 - slope).
141
142     perPoint = step + (iterStep1 - step) / (1 - slope);
143
144     perPointCount = perPointCount + 1;
145     Periodic_points(perPointCount) = perPoint;
146     Slope(perPointCount) = slope;
147
148     if Figures > 0
149         fprintf('At %11.10f is a', perPoint)
150
151     if abs(slope) < 1
152         fprintf('    STABLE periodic point. (slope = %8.7f).\n',
153             slope)
154
155         % Plot stable period point interval green.
156
157         plot(perPoint, perPoint, "g*")
158     else
159         fprintf('n unstable periodic point. (slope = %8.7f).\n',
160             slope)
161
162         % Plot unstable period point interval red.

```



```

162         plot(perPoint, perPoint, "r.")
163     end
164 end
165 end
166 end
167
168
169 if Figures > 0
170 title(['The map iterated ', num2str(Composition_fold), ' times'])
171 xlabel('M_n')
172 ylabel(['M_{n+}', num2str(Composition_fold), ''])
173
174 hold off
175 end

```

TMperiod2.m

```

1 function [] = TMperiod2(Map, Number_of_points, Max_iter, ...
2     Epsilon_array, Dot_size)
3
4 % Toy Model
5 %
6 % This function generates periodic points of a map.
7 %
8 % Map            Enter a function handle to evaluate.
9 %
10 % Number_of_points The number of points checked for periodicity.
11 %
12 % Max_iter        The maximum number of map iteration to determine
13 %                  whether or not a point is (close to be) a periodic
14 %                  point.
15 %
16 % Epsilon_array    An array of epsilon values. The period of the first
17 %                  iterate that has a difference smaller than epsilon,
18 %                  for each epsilon in the array, will be saved.
19
20 clf
21 close all
22
23 mapPlot = figure;
24
25 % Check if Initial_value, Number_of_iter, Plot_until_iter and Map_iter
26 % have valid values.
27
28 if (Max_iter <= 0) || (Max_iter ~= round(Max_iter))
29     disp('! No positive integer input into variable "Max_iter_dense" of
30         function "TM".')
31 end
32
33 Epsilon_array = sort(Epsilon_array);

```

```

34 % Choosing a map and plotting it.
35
36 figure(mapPlot);
37
38 hold on
39
40 if isa(Map, 'function_handle')
41     map = Map;
42     x = 0: .001: 1; % This map is not directly plotted like
43     y = zeros(1, 1001); % the previous two. The y coordinates
44     for it =1: 1001 % are generated using its map function.
45         y(it)= map(x(it));
46     end
47     plot(x,y)
48
49 else
50     disp('! No function handle input into variable "Map" of function "TM
51         "."')
52
53 end
54
55 hold off
56
57 solArray = zeros( length(Epsilon_array)+1, Number_of_points);
58
59 solArray(1,:) = linspace(0, 1, Number_of_points);
60
61 periodicPlot = figure;
62
63 % Since this process can take a long time, a loading bar will
64 % visualize the progress.
65
66 loading = 0;
67 fprintf('\n Finding all periodic points...\n')
68 fprintf(' |0%%-----100%%\n ')
69
70 for iterPoint = 1: Number_of_points
71
72     iterVal = solArray(1, iterPoint);
73
74     for iterMap = 1: Max_iter
75
76         if solArray(length(Epsilon_array) + 1, iterPoint) == 0
77
78             iterVal = Map(iterVal);
79
80             for iterEpsilon = 1: length(Epsilon_array)
81
82                 if solArray(iterEpsilon + 1, iterPoint) == 0
83
84                     if Epsilon_array(length(Epsilon_array) + 1 -
85                         iterEpsilon) ...

```

```

84         > abs(iterVal - solArray(1, iterPoint))
85
86         solArray(iterEpsilon + 1, iterPoint) = iterMap;
87
88         end
89     end
90 end
91 end
92 end
93
94 if loading < floor(30 * iterPoint / Number_of_points)
95
96     loading = floor(30 * iterPoint / Number_of_points);
97     fprintf('#')
98
99 end
100 end
101
102 percentageDense = [sum(solArray(length(Epsilon_array) + 1, :) > 0) * 100
    / Number_of_points sum(solArray(length(Epsilon_array), :) > 0) * 100 /
    Number_of_points];
103
104 fprintf('\n\nThe map, iterated at most %3.0f times, is checked for %6.0f
    initial values.\n', ...
105     Max_iter, Number_of_points)
106 fprintf('The periodic points are %6.4f percent dense, using an epsilon of
    %12.11f.\n', percentageDense(1), Epsilon_array(1, 1))
107 fprintf('The periodic points are %6.4f percent dense, using an epsilon of
    %12.11f.\n', percentageDense(2), Epsilon_array(1, 1))
108
109
110 figure(periodicPlot);
111
112 hold on
113
114 for iterEpsilon = 1: length(Epsilon_array)
115
116     scatter(solArray(1, :), solArray(iterEpsilon + 1, :), Dot_size, ...
117         [min(1 - (iterEpsilon - 1) / (length(Epsilon_array) - 1),
118             solArray(iterEpsilon + 1, :)).' min((iterEpsilon - 1) / (
119                 length(Epsilon_array) - 1), solArray(iterEpsilon + 1, :)).' 1
120             - min(1, solArray(iterEpsilon + 1, :)).')]
121
122     % [1 - min(1, solArray(iterEpsilon + 1, :)).' zeros(1,
123         Number_of_points).' min(1, solArray(iterEpsilon + 1, :)).']
124
125 end
126
127 set(gca, 'yscale', 'log')
128 title('The period of (close to) periodic points')
129 xlabel('Values')
130 ylabel(['Iterations to get within epsilon = ', num2str(Epsilon_array(1,

```

```

127     1)))
128 end

```

TMbifur.m

```

1 function [] = TMbifur(Map, Steps, Min_mu, Max_mu,...
2     Min_value, Max_value, Transient, Max_iter, Initial,...
3     Dot_size, Last_out_first_in, Keep_figures)
4
5 % TM bifur
6 %
7 % Create a bifurcation diagram for a 2-parameter map.
8 %
9 % Map                Enter a 2-parameter function handle to evaluate.
10 %
11 % Min_mu             The minimum value of the map's second parameter.
12 %
13 % Max_mu             The maximum value of the map's second parameter.
14 %
15 % Steps              The number of steps used for second parameter values
16 %                    between Min_mu and Max_mu.
17 %
18 % Transient          The number of iterations that will be omitted from
19 %                    the results for the values at a given mu. When this
20 %                    is set to -1, the initial value will be added.
21 %
22 % Max_iter           The number of iterations used to gather values.
23 %
24 % Initial            The initial value used to start the iterations.
25 %
26 % Dot_size           The size of the dots in the bifurcation diagram.
27 %
28 % Last_out_first_in  When this is set to 1, the last iterate of a value
29 %                    of mu will be the initial value for the next mu.
30 %
31 % Keep_figures       When this is set to 1, the previous figures will not
32 %                    be closed.
33 %
34 % In the case that some parameters were not filled in, standard values
35 % will be applied.
36
37 if nargin <= 8
38     Initial = 0;
39 end
40
41 if nargin <= 9
42     Dot_size = .5;
43 end
44
45 if nargin <= 10

```

```

46     Last_out_first_in = 1337;
47 end
48
49 if nargin <= 11
50     Keep_figures = 1337;
51 end
52
53 if Keep_figures ~= 1
54     clf
55     close all
56 end
57
58 mapPlot = figure;
59 bifurPlot = figure;
60
61 % Check whether the variable is a function handle and plot the function
62 % for the minimum mu and the maximum mu.
63
64 figure(mapPlot);
65 hold on
66
67 if isa(Map, 'function_handle')
68     map = Map;
69     x = 0: .001: 1;
70     y1 = zeros(1, 1001);
71     y2 = y1;
72     y3 = y1;
73
74     for it =1: 1001
75         y1(it)= map(x(it),Min_mu);
76         y2(it)= map(x(it),.5*(Min_mu + Max_mu));
77         y3(it)= map(x(it),Max_mu);
78     end
79
80     plot(x,y1,"b")
81     plot(x,y2,"m")
82     plot(x,y3,"r")
83
84 else
85     disp('! No function handle input into variable "Map" of function "TM
86         "."')
87 end
88
89 % Create a solution array for all iterations of every step between the
90 % minimum mu and maximum mu.
91
92 solArray = zeros(Max_iter - Transient, Steps + 1);
93
94 iterValue = Initial;
95
96 % Display a loading bar to visualize the progress.

```

```

97
98 loading = 0;
99 fprintf( '\n Calculating all points...\n' )
100 fprintf( '|0%%-----100%%|\n ' )
101
102 figure(bifurPlot);
103
104 hold on
105
106 for step = 0: Steps
107
108     mu = Min_mu + (Max_mu - Min_mu) * step / Steps; % Current mu.
109
110     % Iterate until just after the transient value.
111
112     for iter = 1: Transient + 1
113
114         iterValue = map(iterValue, mu);
115     end
116
117     % Let the first value after the transient be the first of the array.
118
119     solArray(1, step + 1) = iterValue;
120
121     % Fill the array with the remaining values.
122
123     for iter = 2: Max_iter - Transient
124
125         solArray(iter, step + 1) = map(solArray(iter - 1, step + 1), mu);
126     end
127
128     % Create a vector of the current step and remove unwanted values.
129
130     solVector = solArray(:, step + 1);
131     solVector = solVector(solVector >= Min_value & solVector <= Max_value
132         );
133
134     % Plot the values of the solution vector with the current mu.
135
136     scatter(ones(size(solVector)) * mu, solVector, Dot_size, 'b')
137
138     % Enlarge the loading bar, depending on the number of steps taken.
139
140     if loading < floor(30 * step / Steps)
141
142         loading = floor(30 * step / Steps);
143         fprintf( '#')
144     end
145
146     if Last_out_first_in ~= 1 % Check whether we want to reset initial.
147
148         iterValue = Initial; % Reset initial value for the next step.

```

```

148     end
149 end
150 fprintf( '\n Plotting ... \n' )
151
152 hold off
153
154 title( 'Bifurcation ' )
155 xlabel( 'Mu' )
156 ylabel( 'Values ' )

```

TMlya_init.m

```

1 function [] = TMlya_init(Map, Derivative, Transient, Max_iter, ...
2     Min_initial, Max_initial, Initial_steps)
3
4 % TM ly a init
5 %
6 % This function calculates the Lyapunov exponent of finite time of
7 % the function over a range of initial conditions.
8 %
9 % Map                Enter a function handle to evaluate.
10 %
11 % Derivative         Enter a function handle that is the derivative of
12 %                   the previous function.
13 %
14 % Transient          The number of iterations that will be omitted from
15 %                   the results for the values at a given mu. When this
16 %                   is set to -1, the initial value will be added.
17 %
18 % Max_iter           The number of iterations used to gather values.
19 %
20 % Min_initial        The smallest initial value used for the iterations.
21 %
22 % Max_initial        The largest initial value used for the iterations.
23 %
24 % Initial_steps      The number of step used for the initial values.
25
26 clf
27 close all
28
29 mapPlot = figure;
30 ly aPlot = figure;
31
32 % Check whether the variable is a function handle and plot the function
33 % for the minimum mu and the maximum mu.
34
35 figure(mapPlot);
36 tiledlayout(1, 2)
37 nexttile
38
39 hold on

```

```

40
41 if isa(Map, 'function_handle')
42     map = Map;
43     x = 0: .001: 1;
44     y = zeros(1, 1001);
45
46     for it =1: 1001
47         y(it)= map(x(it));
48     end
49
50     plot(x,y,"b")
51
52 else
53     disp('! No function handle input into variable "Map" of function "
54         TMlya_init".')
55 end
56 hold off
57
58 % The same for the derivative.
59
60 nexttile
61
62 hold on
63
64 if isa(Map, 'function_handle')
65     dmap = Derivative;
66     x = 0: .001: 1;
67     y = zeros(1, 1001);
68
69     for it =1: 1001
70         y(it)= dmap(x(it));
71     end
72
73     plot(x,y,"r")
74
75 else
76     disp('! No function handle input into variable "Derivative" of
77         function "TMlya_init".')
78 end
79
80 % Set standard values if a parameter was left empty.
81
82 if nargin <= 2
83     Transient = 1000;
84 end
85
86 if nargin <= 3
87     Max_iter = Transient + 500;
88 end
89

```



```

90 if nargin <= 4
91     Min_initial = 0;
92 end
93
94 if nargin <= 5
95     Max_initial = 1;
96 end
97
98 if nargin <= 6
99     Initial_steps = 100;
100 end
101
102 % Create a Lyapunov exponent array large enough for every step between
103 % the minimum initial value and the maximum initial value.
104
105 leArray = zeros(1, Initial_steps + 1);
106
107 % Display a loading bar to visualize the progress.
108
109 loading = 0;
110 fprintf('\n Calculating all Lyapunov exponents...\n')
111 fprintf('|0%%-----100%%|\n ')
112
113 for step = 0: Initial_steps % Use precision for maximum steps.
114
115     % Calculate the initial value of the current step.
116
117     iterValue = Min_initial + (Max_initial - Min_initial) * step /
        Initial_steps;
118
119     % Iterate until just after the transient value.
120
121     for iter = 1: Transient
122
123         iterValue = map(iterValue);
124     end
125
126     % Fill the array with the remaining values.
127
128     for iter = 1: Max_iter - Transient
129
130         iterValue = map(iterValue);
131
132         % Add the derivative of that iterated value divided by the total
133         % number of iterations to the Lyapunov exponent array.
134
135         leArray(step + 1) = leArray(step + 1) + log(abs(Derivative(
            iterValue))) / (Max_iter - Transient);
136     end
137
138     % Enlarge the loading bar, depending on the number of steps taken.
139

```

```

140     if loading < floor(30 * step / Initial_steps)
141
142         loading = floor(30 * step / Initial_steps);
143         fprintf('#')
144     end
145
146 end
147
148 fprintf('\n Plotting ... \n')
149
150 figure(lyaPlot);
151
152 hold on
153
154 plot(linspace(Min_initial, Max_initial, Initial_steps + 1), leArray * 0,
      "k")
155 plot(linspace(Min_initial, Max_initial, Initial_steps + 1), leArray, "b")
156
157 hold off
158
159 title(['Lyapunov exponents of finite time (n goes from ', num2str(
      Transient + 1), ' to ', num2str(Max_iter), ').'])
160 xlabel('Initial values')
161 ylabel('Lyapunov exponents')

```

TMlya_mu.m

```

1 function [] = TMlya_mu(Map, Derivative, Min_mu, Max_mu, Mu_steps,...
2     Transient, Max_iter, Min_initial, Max_initial, Initial_steps)
3
4 % TM lya mu
5 %
6 % This function calculates the minimum, mean and maximum (depending on
7 % the initial values) Lyapunov exponent of finite time of the function
8 % over a range of mu values.
9 %
10 % Map                Enter a 2-parameter function handle to evaluate.
11 %
12 % Derivative          Enter a 2-parameter function handle that is the
13 %                     derivative of the previous function.
14 %
15 % Min_mu              The minimum value of the map's second parameter.
16 %
17 % Max_mu              The maximum value of the map's second parameter.
18 %
19 % Mu_steps            The number of steps used for second parameter values
20 %                     from Min_mu to Max_mu.
21 %
22 % Transient           The number of iterations that will be omitted from
23 %                     the results for the values at a given mu. When this
24 %                     is set to -1, the initial value will be added.

```

```

25 %
26 % Max_iter          The number of iterations used to gather values.
27 %
28 % Min_initial       The smallest initial value used for the iterations.
29 %
30 % Max_initial       The largest initial value used for the iterations.
31 %
32 % Initial_steps     The number of steps used for the initial values.
33
34 clf
35 close all
36
37 mapPlot = figure;
38 lyaPlot = figure;
39
40 % Check whether the variable is a function handle and plot the function
41 % for the minimum mu and the maximum mu.
42
43 figure(mapPlot);
44 tiledlayout(1, 2)
45 nexttile
46
47 hold on
48
49 if isa(Map, 'function_handle')
50     map = Map;
51     x = 0: .001: 1;
52     y1 = zeros(1, 1001);
53     y2 = y1;
54     y3 = y1;
55
56     for it =1: 1001
57         y1(it)= map(x(it),Min_mu);
58         y2(it)= map(x(it),.5*(Min_mu + Max_mu));
59         y3(it)= map(x(it),Max_mu);
60     end
61
62     plot(x,y1,"b")
63     plot(x,y2,"m")
64     plot(x,y3,"r")
65
66 else
67     disp('! No function handle input into variable "Map" of function "
        TMlya_mu".')
68
69 end
70
71 hold off
72
73 % The same for the derivative.
74
75 nexttile

```

```

76
77 hold on
78
79 if isa(Map, 'function_handle')
80     dmap = Derivative;
81     x = 0: .001: 1;
82     y1 = zeros(1, 1001);
83     y2 = y1;
84     y3 = y1;
85
86     for it =1: 1001
87         y1(it)= dmap(x(it),Min_mu);
88         y2(it)= dmap(x(it),.5*(Min_mu + Max_mu));
89         y3(it)= dmap(x(it),Max_mu);
90     end
91
92     plot(x,y1,"b")
93     plot(x,y2,"m")
94     plot(x,y3,"r")
95
96 else
97     disp('! No function handle input into variable "Derivative" of
98         function "TMlya_mu".')
99 end
100
101 % Set standard values if a parameter was left empty.
102
103 if nargin <= 4
104     Mu_steps = 100;
105 end
106
107 if nargin <= 5
108     Transient = 1000;
109 end
110
111 if nargin <= 6
112     Max_iter = Transient + 500;
113 end
114
115 if nargin <= 7
116     Min_initial = 0;
117 end
118
119 if nargin <= 8
120     Max_initial = 1;
121 end
122
123 if nargin <= 9
124     Initial_steps = 100;
125 end
126

```

```

127 % Create a mu array large enough for every step between the minimum mu
128 % and the maximum mu. Every mu will get a minimum, average and maximum
129 % Lyapunov exponent, since these are dependent on the initial values.
130
131 muArray = zeros(3, Mu_steps + 1);
132
133 % Display a loading bar to visualize the progress.
134
135 loading = 0;
136 fprintf('\n Calculating all Lyapunov exponents...\n')
137 fprintf('|0%%-----100%%|\n ')
138
139 for stepMu = 0: Mu_steps
140
141     mu = Min_mu + (Max_mu - Min_mu) * stepMu / Mu_steps;% Current mu.
142
143     % Create a Lyapunov exponent array large enough for every step
144     % between the minimum initial value and the maximum initial value.
145
146     leArray = zeros(1, Initial_steps + 1);
147
148     for stepInit = 0: Initial_steps
149
150         % Calculate the initial value of the current step.
151
152         iterValue = Min_initial + (Max_initial - Min_initial) * stepInit
            / Initial_steps;
153
154         % Iterate until just after the transient value.
155
156         for iter = 1: Transient
157
158             iterValue = map(iterValue, mu);
159         end
160
161         % Fill the array with the remaining values.
162
163         for iter = 1: Max_iter - Transient
164
165             iterValue = map(iterValue, mu);
166
167             % Add the derivative of that iterated value divided by the
                total
168             % number of iterations to the Lyapunov exponent array.
169
170             leArray(stepInit + 1) = leArray(stepInit + 1) + log(abs(
                Derivative(iterValue, mu))) / (Max_iter - Transient);
171         end
172     end
173
174     muArray(:, stepMu + 1) = [min(leArray); mean(leArray); max(leArray)];
175

```

```

176     % Enlarge the loading bar, depending on the number of evaluated mu's.
177
178     if loading < floor(30 * stepMu / Mu_steps)
179
180         loading = floor(30 * stepMu / Mu_steps);
181         fprintf('#')
182     end
183 end
184
185 fprintf('\n Plotting...\n')
186
187 figure(lyaPlot);
188
189 hold on
190
191 plot(linspace(Min_mu, Max_mu, Mu_steps + 1), muArray(1,:) * 0, "k")
192 plot(linspace(Min_mu, Max_mu, Mu_steps + 1), muArray(1,:), "g")
193 plot(linspace(Min_mu, Max_mu, Mu_steps + 1), muArray(2,:), "b")
194 plot(linspace(Min_mu, Max_mu, Mu_steps + 1), muArray(3,:), "r")
195
196 hold off
197
198 title(['Lyapunov exponents of finite time (n goes from ', num2str(
199     Transient + 1), ' to ', num2str(Max_iter), ').'])
200 xlabel('Mu')
201 ylabel({'Lyapunov exponents over initial values', '(min = green, mean =
    blue, max = red)'})

```

TMmulti.m

```

1 function [] = TMmulti(Map, Initials_array, Number_of_iter)
2
3 % Toy Model multi
4
5 % Creates multiple graphical analyses in one figure.
6 %
7 % Map                Enter a function handle to evaluate.
8 %
9 % Initials_array     An array of initial values for each analysis.
10 %
11 % Number_of_iter     The number of iterations for the analyses.
12
13
14 close all
15
16 figure;
17
18 hold on
19
20 x = 0: .001: 1;                % This map is not directly plotted like
21 y = zeros(1, 1001);           % the previous two. The y coordinates

```

```

22 for it =1: 1001                                % are generated using its map function.
23     y(it)= Map(x(it));
24 end
25 plot(x,y)
26
27 initialsLength = length(Initials_array);
28
29 solArray = zeros(Number_of_iter + 1, initialsLength); % Generate a large
30     enough array.
31 solArray(1, 1:initialsLength) = Initials_array;    % Let initial value be
32     the first value.
33
34 plot([0, 1], [0, 1], "k");                      % Plot the identity line (black).
35
36 for itleng = 1: initialsLength
37     for iter = 1: Number_of_iter
38
39         solArray(iter + 1, itleng) = Map(solArray(iter , itleng));    %
40             Filling solArray
41
42         plot([solArray(iter , itleng), solArray(iter , itleng)],... % The
43             line from the Map
44             [solArray(iter , itleng), solArray(iter+1, itleng)],... % to
45             the identity Map.
46             'color', [iter/Number_of_iter 1-iter/Number_of_iter 0])
47
48         plot([solArray(iter , itleng), solArray(iter+1, itleng)],... % The
49             line from the id
50             [solArray(iter+1, itleng), solArray(iter+1, itleng)],...% Map
51             to the next point.
52             'color', [iter/Number_of_iter 1-iter/Number_of_iter 0])
53
54     end
55 end
56 title(['The iterated value from initial (green) to the ',...
57     num2str(Number_of_iter), '-th (red)'])
58 xlabel('Value N')
59 ylabel('Value N+1')
60
61 hold off

```

TMcompmovie.m

```

1 function [] = TMcompmovie(Map, Composition_fold,...
2     Min_mu, Max_mu, Number_of_mus, Precision, Min_x, Max_x, Min_y, Max_y)
3
4 % Toy Model composition
5 %
6 % This function creates an animation of the N-fold composition of a
7 % 2-parameter map, shifting the second value through time. Periodic
    points

```

```

8 % are shown with red dots if they are unstable and with a green star if
9 % they are stable.
10 %
11 % Map          Enter a 2-parameter function handle to evaluate.
12 %
13 % Composition_fold  The considered composition of the map.
14 %
15 % Min_mu, Max_mu    The range of the second parameter of the map.
16 %
17 % Number_of_mus     The division of the previous mentioned range.
18 %
19 % Min_x,... , Max_y  The range used for the animation.
20
21 close all
22
23 % Set parameters
24
25 iterPlot = figure;
26
27 muArray = linspace(Min_mu, Max_mu, Number_of_mus);
28 stepsArray = 0: Precision: 1;
29
30 % Plot the n-th iteration of the map and find its periodic points.
31
32 figure(iterPlot);
33
34 for iterMu = 1: Number_of_mus
35
36     clf
37
38     hold on
39
40     plot([0, 1], [0, 1], "k")          % Plot the identity map in black.
41
42     for step = 1: size(stepsArray,2) - 1
43         iterStep1 = stepsArray(step);
44         iterStep2 = stepsArray(step + 1);
45
46         for mapIter = 1: Composition_fold
47             iterStep1 = min(1, max(0, Map(iterStep1, muArray(iterMu))));
48             iterStep2 = min(1, max(0, Map(iterStep2, muArray(iterMu))));
49         end
50
51         plot([stepsArray(step), stepsArray(step+1)], [iterStep1,
52             iterStep2], "b")
53
54         if (iterStep1 - stepsArray(step)) * (iterStep2 - stepsArray(step
55             +1)) <= 0
56
57             slope = (iterStep2 - iterStep1) / Precision;
58             perPoint = stepsArray(step) + (iterStep1 - stepsArray(step))
59                 / (1 - slope);

```



```

57         if abs(slope) < 1
58             % Plot stable period point interval green.
59             plot(perPoint, perPoint, "g*")
60         else
61             % Plot unstable period point interval red.
62             plot(perPoint, perPoint, "r.")
63         end
64     end
65 end
66
67 xlim([Min_x Max_x])
68 ylim([Min_y Max_y])
69
70
71 title(['The map iterated ', num2str(Composition_fold), ' times'])
72 xlabel('M_n')
73 ylabel(['M_{n+}', num2str(Composition_fold), '}'])
74
75 compMovie(iterMu) = getframe;
76 end
77
78 movieWriter = VideoWriter('Comp_Movie_10_0001', 'MPEG-4');
79 movieWriter.FrameRate = 10;
80
81 open(movieWriter);
82 writeVideo(movieWriter, compMovie);
83 close(movieWriter);

```

TMstableset.m

```

1 function [] = TMstableset(Map, Max_composition, Precision, Set_size,
2     Set_iter_max, Dot_size)
3
4 % Toy Model composition
5 % This function shows unstable set of every repelling periodic point.
6 %
7 % Map          Enter a function handle to evaluate.
8 %
9 % Max_composition  If this number is non-zero, it will find the
10 %                periodic points of equal or less than this number
11 %                of iterations for the choosen map.
12 %
13 % Precision      The precision used for calculating periodic points.
14 %                Use a decimal number to determine the step size.
15 %
16 % Set_size       The number of elements for every unstable set.
17
18 solArray = NaN(2^(Max_composition + 1) - 2 - Max_composition, Set_size);
19 xArray = solArray;
20 compPerPntPosition = 0;

```

```

21 initialValues = linspace(0, 1, Set_size);
22
23 % Display a loading bar to visualize the progress.
24
25 fprintf( '\n Going through all compositions ... \n' )
26 fprintf( '|1--4--7--10--14--18|\n ' )
27
28 for composition = 1: Max_composition
29
30     [perPoints, slope] = TMcomposition(Map, composition, Precision, 0, 0)
31     ;
32
33     for itPoint = 1: size(perPoints,1)
34
35         if abs(slope(itPoint)) < 1
36
37             solArray(compPerPntPosition + itPoint, 1) = ...
38                 perPoints(itPoint);
39             xArray(compPerPntPosition + itPoint, :) = ...
40                 repmat(perPoints(itPoint), 1, Set_size);
41
42             for itSet = 1: Set_size
43
44                 setIter = 0;
45                 value = initialValues(itSet);
46
47                 while setIter < Set_iter_max &&...
48                     isnan(solArray(compPerPntPosition + itPoint,
49                                     itSet))
50
51                     value = Map(value);
52
53                     if abs(perPoints(itPoint) - value) < Precision
54
55                         solArray(compPerPntPosition + itPoint, itSet) =
56                             initialValues(itSet);
57                     end
58
59                     setIter = setIter + 1;
60                 end
61             end
62         end
63
64         compPerPntPosition = compPerPntPosition + size(perPoints,1);
65
66         fprintf( '#' )
67
68     end
69
70     fprintf( '\nPlotting ... \n' )
71
72     scatter(reshape(xArray, [], 1), reshape(solArray, [], 1), Dot_size, 'b')

```

```

70 title(['The stable sets of per. points with max. period ', num2str(
    Max_composition)])
71 xlabel('Values of existing stable per. points')
72 ylabel(['The set converging to per. point within ', num2str(Set_size), '
    iterations'])
73 xlim([0 1])
74 ylim([0 1])

```

TMunstableset.m

```

1 function [] = TMunstableset(Map, Max_composition, Precision, Epsilon,
    Set_size, Dot_size)
2
3 % Toy Model composition
4 %
5 % This function shows unstable set of every repelling periodic point.
6 %
7 % Map          Enter a function handle to evaluate.
8 %
9 % Max_composition If this number is non-zero, it will find the
10 %                periodic points of equal or less than this number
11 %                of iterations for the choosen map.
12 %
13 % Precision      The precision used for calculating periodic points.
14 %                Use a decimal number to determine the step size.
15 %
16 % Set_size       The number of elements for every unstable set.
17
18 solArray = NaN(2^(Max_composition + 1) - 2 - Max_composition, Set_size);
19 xArray = solArray;
20 compPerPntPosition = 0;
21
22 % Display a loading bar to visualize the progress.
23
24 fprintf('\n Going through all compositions...\n')
25 fprintf('|1--4--7--10--14--18|\n ')
26
27 for composition = 1: Max_composition
28
29     [perPoints, slope] = TMcomposition(Map, composition, Precision, 0, 0)
30     ;
31
32     solArray(compPerPntPosition + 1: compPerPntPosition + size(perPoints
33     ,1), 1) =...
34     perPoints + Epsilon;
35     xArray(compPerPntPosition + 1: compPerPntPosition + size(perPoints,1)
36     , :) =...
37     repmat(perPoints, 1, Set_size);
38
39     for itPoint = 1: size(perPoints,1)

```

```

38         if abs(slope(itPoint)) >= 1
39
40             for itSet = 2: Set_size
41
42                 solArray(compPerPntPosition + itPoint, itSet) = ...
43                 Map(solArray(compPerPntPosition + itPoint, itSet - 1)
44                     );
45             end
46         end
47
48         compPerPntPosition = compPerPntPosition + size(perPoints,1);
49
50         fprintf('#')
51     end
52
53     fprintf('\nPlotting...\n')
54
55     scatter(reshape(xArray, [], 1), reshape(solArray, [], 1), Dot_size, 'b')
56     title(['The unstable sets of per. points with max. period ', num2str(
57         Max_composition)])
58     xlabel('Values of existing unstable per. points')
59     ylabel(['The unstable set within ', num2str(Set_size), ' iterations'])
60     xlim([0 1])
61     ylim([0 1])

```

TMpredictability.m

```

1 function [valsArray,ecount] = TMPredictability(Map, Tau, Sample_size,...
2     Event_limit, Initial, Map_precision, Want_plot)
3
4 % Calculate the time-Tau Lyapunov exponent for extreme events of a map.
5 %
6 % Map           Enter a function handle to evaluate.
7 %
8 % Tau           The lead time.
9 %
10 % Sample_size   The number of iterations of the forward orbit to
11 %               investigate.
12 %
13 % Event_limit   The value for which exceeding values are considered
14 %               as extreme events.
15 %
16 % Initial       The initial value.
17 %
18 % Map_precision Precision used for calculating the slope of the map.
19 %
20 %
21
22 close all
23

```

```

24 orbitArray = NaN(1, Sample_size);
25 valsArray = NaN(3, Sample_size);
26 orbitArray(1,1) = Initial;
27 ecount = 0;
28
29 mapPoints = 0 : Map_precision : 1;
30 mapPoints = [mapPoints ; mapPoints];
31 [~, S] = size(mapPoints);
32 for iter = 1 : S
33     [~, p] = dfunc(Map, Tau, mapPoints(1, iter), Map_precision);
34     mapPoints(2, iter) = p;
35 end
36
37 for iter = 1 : Sample_size - 1
38     orbitArray(1, iter + 1) = Map(orbitArray(1, iter));
39
40     if Tau <= iter && orbitArray(1, iter + 1) > Event_limit
41         ecount = ecount + 1;
42         valsArray(1, ecount) = orbitArray(1, iter + 1 - Tau);
43     end
44 end
45
46 valsArray = valsArray(:, 1 : ecount);
47
48 for iter = 1 : ecount
49     [slope, vals] = dfunc(Map, Tau, valsArray(1, iter), Map_precision);
50     valsArray(2 : 3, iter) = [vals, log(abs(slope))/Tau].';
51 end
52
53 [arrayCount, arrayVals] = groupcounts(round(valsArray(3, :).', -floor(
    log10(Map_precision))));
54
55 if Want_plot == 1
56
57     disp(arrayCount.')
58     disp(arrayVals.')
59     disp(mean(valsArray(3, :)))
60     disp(sum(valsArray(3, :)<0)/ecount)
61     %disp(ecount)
62     %disp(sum(valsArray(3, :)<0))
63     hold on
64     plot(mapPoints(1, :), mapPoints(2, :))
65     scatter(valsArray(1, :), valsArray(2, :), 'r.')
66     xlabel('value x')
67     ylabel([num2str(Tau), '-fold map applied to x'])
68     title(['Visualization of extreme events after lead time ', num2str(Tau)])
69     hold off
70     figure
71     xlim([0 1])
72     scatter(valsArray(1, :), valsArray(3, :), 'r.')
73     xlabel('value x')
74     ylabel(['time-', num2str(Tau), ' Lyapunov exponent of the ', num2str(Tau), '-

```

```

    fold map applied to x'])
75 title(['Lyapunov exponents of extreme events after lead time ', num2str(
    Tau)])
76 figure
77 histogram(valsArray(3, :), floor(min(valsArray(3, :))):ceil(max(valsArray
    (3, :))))
78 xlabel('values of Lyapunov exponents')
79 ylabel('occurrences')
80 title('Histogram of Lyapunov exponents')
81 end
82
83
84
85 function [slope, x] = dfunc(Map, Comp, Initial, Precision)
86
87 % d func
88 %
89 % Calculate the slope of the n-fold composition of a map at an initial.
90
91 if isa(Map, 'function_handle')
92
93 else
94     disp('! No function handle input into variable "Map" of function "
        dfunc".')
95 end
96
97 x = Initial;
98
99 slope = 1;
100
101 for it = 1: Comp
102
103     slope = slope * (Map(x + Precision) - Map(x)) / ...
104         (Precision);
105
106     x = Map(x);
107
108 end

```

Index

- backward asymptotic, 11
- backward orbit, 9
- bifurcation diagram, 16
- bijective, 9
- chaos
 - in the sense of Devaney, 14
 - in the sense of Marotto, 15
- co-domain, 9
- continuously differentiable, 14
- differentiable, 12
- domain, 9
- dynamical system, 8
 - continuous, 8
 - dimension, 8
 - discrete, 8
- El Niño, 5
 - characteristics, 24
 - map, 22
 - maximum period, 24
 - period, 22
 - point, 22
- ENSO model, 6
- expanding fixed point, 14
 - regular, 14
- fixed point, 10
- forward asymptotic, 11
- forward orbit, 9
- graphical analysis, 10
- hyperbolic periodic point, 12
- image, 9
- initial condition, 8
- intermediate value theorem, 47
- invertible, 9
- local stable set, 13
- local unstable set, 13
- Lyapunov exponent, 13
- map, 9
- mean value theorem, 12
- multiplier, 12
- n-fold composition, 9
- neighbourhood, 14
- nth iterate, 9
- one-to-one, 9
- onto, 9
- partial inverse, 9
- periodic point, 10
 - attracting, 11
 - period, 10
 - prime period, 10
 - repelling, 11
- phase space, 8
- Poincaré map, 10
- Poincaré section, 10
- pre-image, 9
- prognostic equation, 8
- smooth, 15
- snap-back repeller, 14
 - non-degenerate, 15
- solution curve, 8
- stable set, 11
- time series, 9
- time-n Lyapunov exponent, 13
- trajectory, 8
- transient, 16
- unstable set, 11