Modelling Chaos
Study and Simulation
of Bifurcation Systems

Boris Ivanov

College of Science
Swansea University

Department of Physics 19th March 2020

Abstract

The purpose of this work is to present the reader with an introduction to linear chaotic systems. We conduct a study of deterministic chaos, proving that the one-dimensional map $f^{n+1}(x_n) = \mu\, x_n\, (1 - x_n)$ meets the requirements for a chaotic system defined by Devaney. Further into the work we obtain Feigenbaums constant using 3 different methods, with the best value obtained being $\delta = 4.6692016$. The data has been generated using independently written software. The software utilises techniques from multiprocessing in order to decrease data processing time, thus allowing for more computationally demanding tasks to be attempted.

Contents

Chapter 1

Introduction

1.1 Chaos as a concept

Chaos
"A state of total confusion with no order."

This is the definition of Chaos given by the Cambridge Dictionary, and a perfect description of the initial impressions one has observing such a system.
Yet physical systems like the double pendulum, vibrating objects, rotating or heated fluids, the motion of a group of celestial objects and dripping faucets are deterministic in nature and can be derived from Newtons laws [Bak96].
Why is it then that we describe them as "total confusion"? To answer this question we look at the definition to a Chaotic System given by Devaney [Dev89]:

Let X be a metric space. A continuous map $f: X \to X$ is said to be chaotic on X if:

1. f is transitive,
2. the periodic points of f are dense in X,
3. f has sensitive dependence on initial conditions.

Each of the three conditions is assured by the existences of the other two [Ban10], [Cra15].
All three are listed for historical reasons and for the strength of the definition, but for practical purposes if two are demonstrated to be valid for a system we will consider it chaotic.
The last condition is perhaps most intuitive. Imagine two systems, one of which starting with slightly different initial conditions to the other. For non-chaotic systems this difference will grow linearly with time. The difference between chaotic systems, however, grows *exponentially* leading to dissimilar behavior very quickly.
Elaborating on the 1st condition. A continuous map or in our case, a dynamical system $f: X \to X$ is said to be transitive if, for any pair U, V of nonempty open subsets of X there exists some positive number of applications $n \geq 0$ of the map f such that $f^n(U) \cap V \neq \emptyset$. [Gro11]
In Layman's terms the above claims that through some n number of applications of f, defined hereafter as f^n, we can reach any point belonging to X. The 2nd rule states that every reachable point belongs to X.

Chaotic systems can be generalised into two families based on the equations that describe them:
1. Linear
2. Non-linear
In our day to day life we predominantly encounter non-linear chaos. Systems being influenced by a myriad of unpredictable and untrackable circumstances. In this work we will study the first kind, which will require us to observe the behaviour of the systems in the long term.

Chapter 2

Proof of Chaotic Behaviour

2.1 Transitivity

For the map $f: X \to X$, where f acts as Eq.(2.1)

$$f^{n+1}(x) = \mu f^n(x) \times (1 - f^n(x)) \tag{2.1}$$

,transitivity means that for the domain X defined by Eq.(2.2),

$$\{x \in \mathbb{R} \mid 0 \leq x \leq 1\} \tag{2.2}$$

there are some finite number of iterations n that would produce a result in the co-domain X defined as Eq.(2.2) ,illustrated in Fig.2.1.

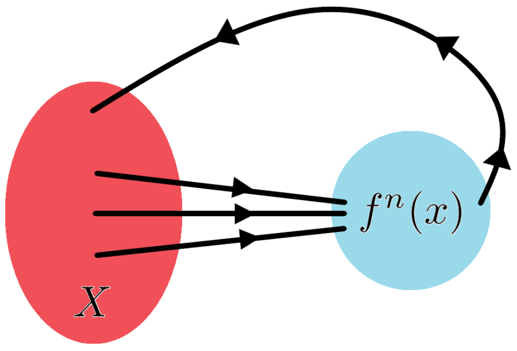

Figure 2.1: Visualisation of $f: X \to X$

This implies that where f behaves chaotically it will **not** converge towards one or multiple values, but rather take up all points within a set. In Fig.2.2 we observe how many unique values f generates against μ, where μ is the Growth Rate as expressed in Eq.(2.1).
Before counting the unique values the map generates, we let the system stabilize running it over 2'000'000 iterations without recording anything. Then values unique to 5 decimal places are recorded. For $0 \leq \mu < 3$, f converges to a single value, then at equal intervals we observe doubling. The system converges to 2, 4, 8, 16, 32, ..., unique values.
As we look at further values of μ (Fig.2.3), this pattern is broken. We stop observing doubling, rather f begins generating a very large amount of unique values. When we begin recording values, we run the map over 50'000 iterations. Out of those, for $\mu > 3.695$, approximately 40'000 are unique to 5 decimal places. This hints that at that Growth Rate f becomes Transitive, generating unique values and allowing for any value in the co-domain to be reached allowing enough iterations. Sec. A.6
The exact value of μ at which Eq.(2.1) becomes Transitive is

$$3.570 \pm 0.005 \tag{2.3}$$

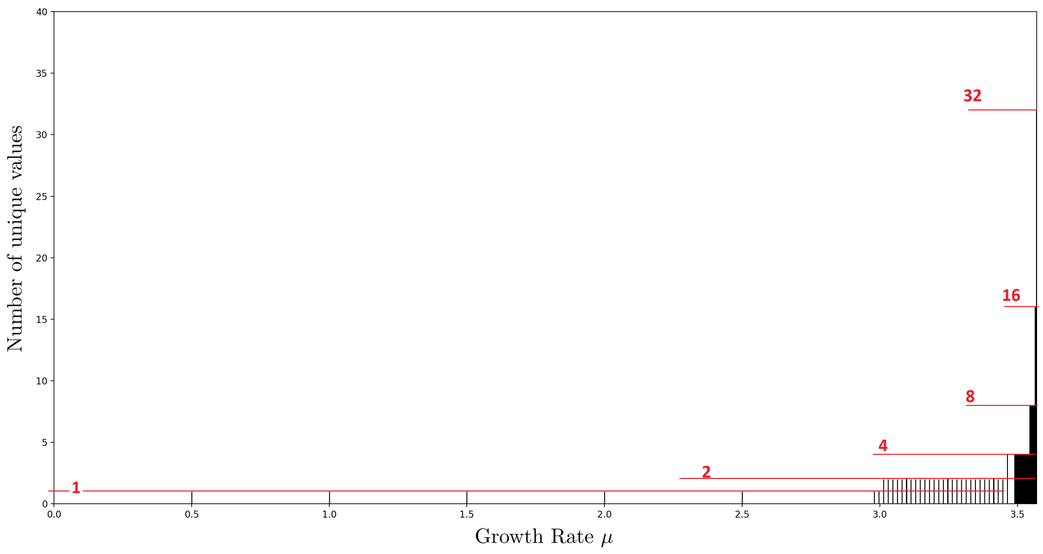

Figure 2.2: Unique values of f to 5 decimal places against μ

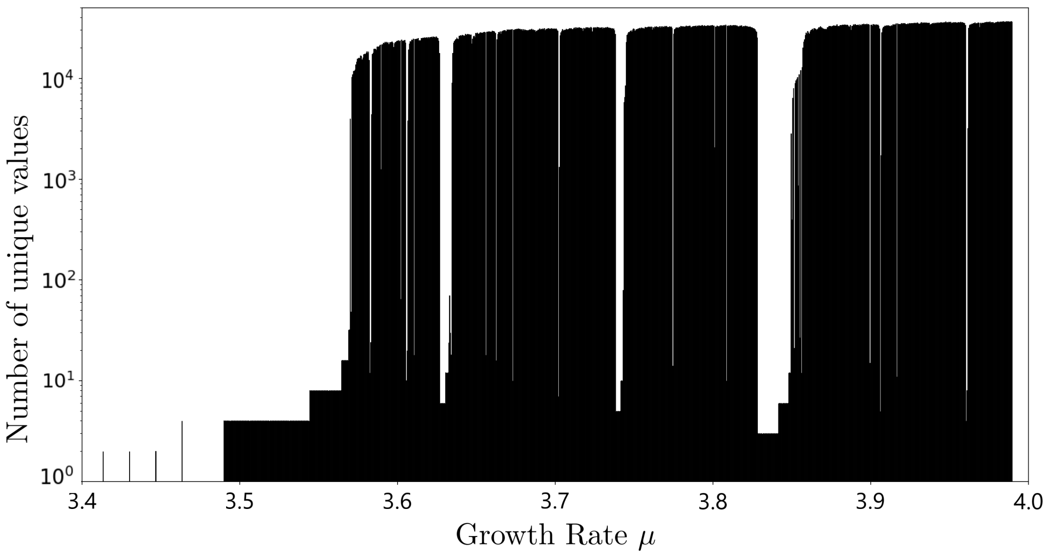

Figure 2.3: Unique values of f to 5 decimal places against μ

2.2 Dense in X

For our map - Eq.(2.1), to be Dense in X, means that all points the map f can generate, regardless of the number of iterations, belong to the domain of X as defined in Eq.(2.2).
While Transitivity and Sensitivity to Initial Conditions generally become true where the map f is chaotic (not surprising, as we are using them as definitions of what being chaotic is), to be Dense in X is a more of a formality. The system being Dense in the co-domain is not a good sign of whether f is chaotic or not. It simply means that the system behaves. For $\mu \in [0,\ 4]$, $f^n(x)$ will always produce a result in the domain, regardless of how many iterations n it undergoes [Bak96] .

Looking at Fig.(2.4) Sec. A.7 we can conclude that Max.$(f^n(x))$ lies at

$$\frac{\partial(f^n(x))}{\partial x} = \mu(1-2x) = 0 \tag{2.4}$$

Thus Max.$(f^n(x))$ resides at $x = 0.5$ for all μ. Using this we can rigorously redefine the domain of μ. $f^n(x) \in [0,\ 1]$; Max.$(f^n(x)) = 1$.

$$MAX\,(f^n(x)) = \mu 0.5(1-0.5) = \frac{\mu}{4} = 1 \tag{2.5}$$

Therefore $\mu = 4$ is the higher boundary, and because negative values are undesirable $\mu = 0$ is the lower boundary.

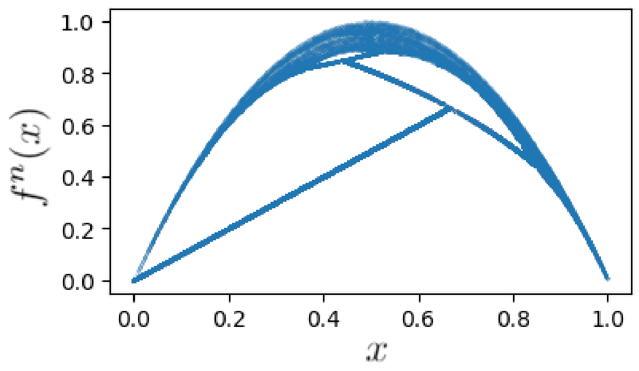

Figure 2.4: Plot of $f^n(x)$ versus x for $\mu \in [0,\ 4]$

What is important is that outside of the scope of μ defined above, the system misbehaves generating negative values or tending to infinity. There are different maps with similar properties, where μ belongs to a different range. For Eq.(2.1) we will only work with $\mu \in [0,\ 4]$, as those are the values where chaos is possible.

2.3 Sensitivity to Initial Conditions

2.3.1 The Lyapunov Exponent

The Lyapunov exponent is a measure of divergence between two systems, $f(x)$ and $f(x + \Delta x)$, evolving from slightly different initial conditions. This divergence is defined to be:

$$f^n(x + \Delta x) - f^n(x) \approx \Delta x\, e^{\lambda n} \tag{2.6}$$

where n is the n^{th} iterate - $f^n(x)$,
λ is the Lyapunov exponent and represents the average rate of divergence.
If λ is negative, $f^n(x)$ and $f^n(x + \Delta x)$ converge as $n \to \infty$.
If λ is positive $f^n(x)$ and $f^n(x + \Delta x)$ diverge; the system is sensitive to initial conditions [Bak96].
To be able to enumerate λ in later computations Eq.((2.6)) is rearranged as shown below.
Eq.((2.6)) can be written as:

$$\lambda \approx \frac{1}{n} ln \left[\frac{f^n(x + \Delta x) - f^n(x)}{\Delta x} \right] \tag{2.7}$$

as $\Delta x \to 0$, this equation takes the form

$$\lambda \approx \frac{1}{n} ln \left| \frac{d(f^n(x))}{dx} \right| \tag{2.8}$$

Applying the chain rule to the derivative of the n^{th} iteration and taking the limit of $n \to \infty$ gives:

$$\lambda = \lim_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n-1} ln|f'(x_i)| \tag{2.9}$$

2.3.2 Methodology and the Code

Throughout this paper there will be several sections discussing the technicalities of the code used to generate the data and graphs used in the writing of this work. The reader should not feel obliged to read through this in detail as the rest of the discussion will repeat any necessary information. This section is meant for those with interest in the exact workings of the software.

While (2.9) is certainly a more mathematically rigorous definition of the Lyapunov exponent, (2.7) has its "derivative" part of the equation expressed in a more code friendly way and ultimately the equation used to obtain the Lyapunov exponent becomes:

$$\lambda = \lim_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n-1} ln \left| \frac{f^n(x_i + \Delta x) - f^n(x_i)}{\Delta x} \right| \tag{2.10}$$

```python
def lyapunov(rate):
    """Returns Lyapunov exponent for rate over N gen."""
    function = 0.2
    g_function = function+DX
    lambda = 0

    for i in range(N):
        function = rate * function * (1 - function)
        g_function = rate * d_function * (1-g_function)

        if g_function-function == 0:
            break

        lambda += d.Decimal(abs(g_function - function) / DX).ln()

    lambda = lambda / (i + 2)
    return lambda
```

Listing 2.1: Lyapunov_Exponent.py excerpt

The **lyapunov** function (as seen in Listing 2.1) begins by defining two variables:

$$f^0(x) = 0.2 \qquad ; \; f^{n+1}(x) = \mu f^n(x) \times (1 - f^n(x)) \tag{2.11}$$

$$g^0(x) = f^0(x) + \Delta x \qquad ; \; g^{n+1}(x) = \mu g^n(x) \times (1 - g^n(x)) \tag{2.12}$$

(2.10) Defines λ with a limit $n \to \infty$, unfortunately that is impractical. Rather than asking **Is it possible to generate an infinite number of iterations?**, we should ask **When is it reasonable to stop?**
To answer this question lets look at where we **have** to stop. The standard *float* has 16 decimals of precision. Importing the DECIMAL module allows pushing that to 26. Pushing that even further, although possible, would heavily impact processing time and is therefore not an option for this project. Even so, if the two systems are identical up to the 26th digit after **n** iterations, it is safe to say that they do not exhibit sensitivity to initial conditions and are therefore not chaotic.
To form an idea of how many iterations it is reasonable to run (Listing 2.1) for, we look at Fig.2.5.

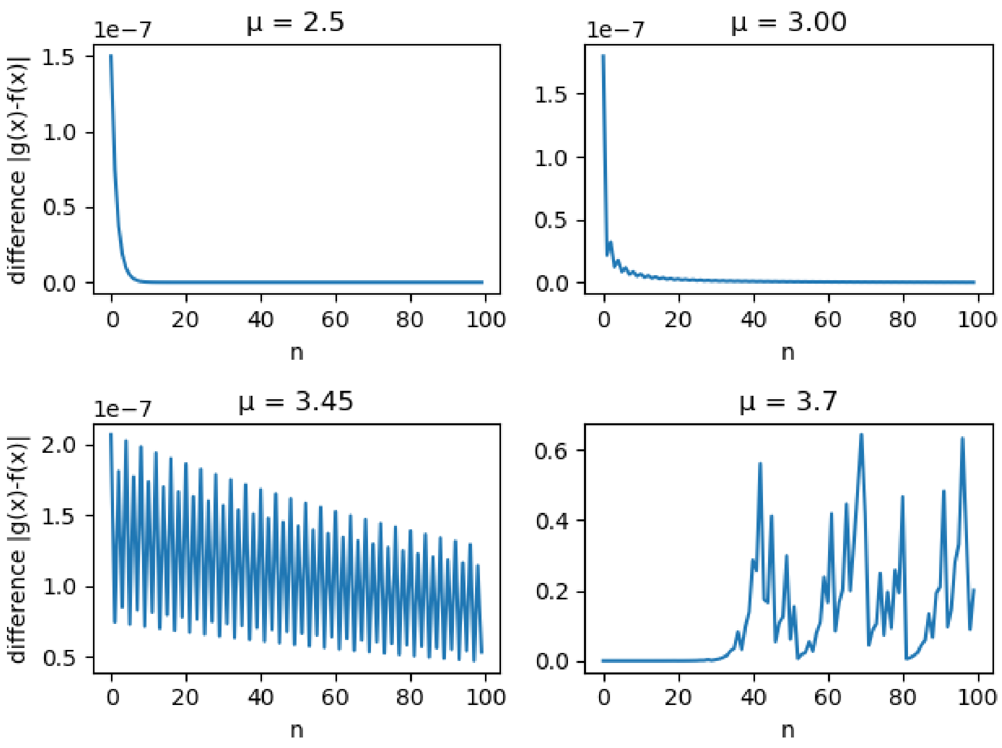

Figure 2.5: Difference $g^n(x) - f^n(x)$ after n-th iteration.

The values for μ specify the rate of growth in $f^{n+1}(x) = \mu f^n(x) \times (1 - f^n(x))$. The paper covers these in detail later. As for now the reader needs to know that:
For $0 < \mu < 1$, $\;f^{n+1}(x) \;\to\; 0$ as $n \to \infty$,
for $1 < \mu < 3$, $\;f^{n+1}(x) \;\to$ const. as $n \to \infty$,

for $3 < \mu < 3.57$, $\quad f^{n+1}(x)$ oscillates between several values
for $\approx 3.57 < \mu < 4$, $\quad f^{n+1}(x)$ becomes chaotic.

Taking another look at Fig.2.5. At low values of growth -μ, it is clear that 20 iterations is enough for the two systems to become nearly identical up to the 26th digit. Where the systems oscillate between values, their difference slowly tends towards 0, which is more obvious in Fig.2.6. 2000 iterations are not needed however, as even after 40 the difference between the two systems is of the order 10^{-7}, which is sufficient to declare them as not sensitive to initial conditions.

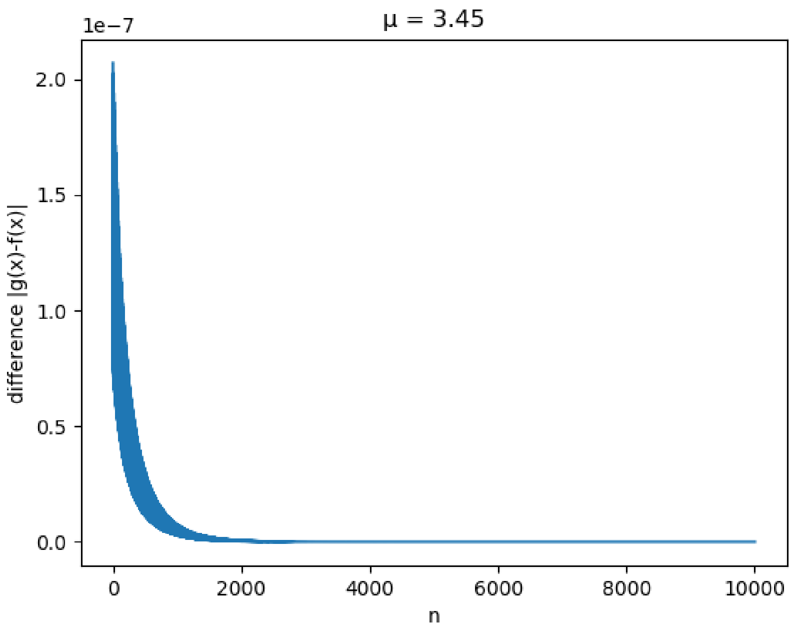

Figure 2.6: Difference $g^n(x) - f^n(x)$ after n-th iteration at rate of growth $\mu = 3.45$. The two systems oscillate between 4 values.

With chaotic systems, interesting behaviour occurs after the 40th iteration exerting the same behaviour to infinity. As such the author finds that iterations over **n=100** are wasteful in time, although some figures are generated with **n=1000**, where higher precision is needed. Equipped with this knowledge we look at (Listing 2.1) again. Iterating $g^n(x)$ and $f^n(x)$ n=100 times, it effectively solves the equation:

$$\lambda = \frac{1}{100} \sum_{i=0}^{99} ln \left| \frac{f^n(x_i + \Delta x) - f^n(x_i)}{\Delta x} \right| \tag{2.13}$$

Unless the difference $g^n(x) - f^n(x) = 0$, in which case it breaks the execution and solves 2.13 for however many times the **for** cycle was iterated. The **lyapunov** function then returns λ and maps it to the corresponding rate (μ) value that it was solved for. The process then start all over again, performing the same operations for a slightly larger rate (μ). In the code as seen at A.2 the difference between one μ and the one for which the **lyapunov** function will be solved thereafter is called RESOLUTION.

2.3.3 Lyapunov Exponent Graph and Sensitivity to Initial Conditions

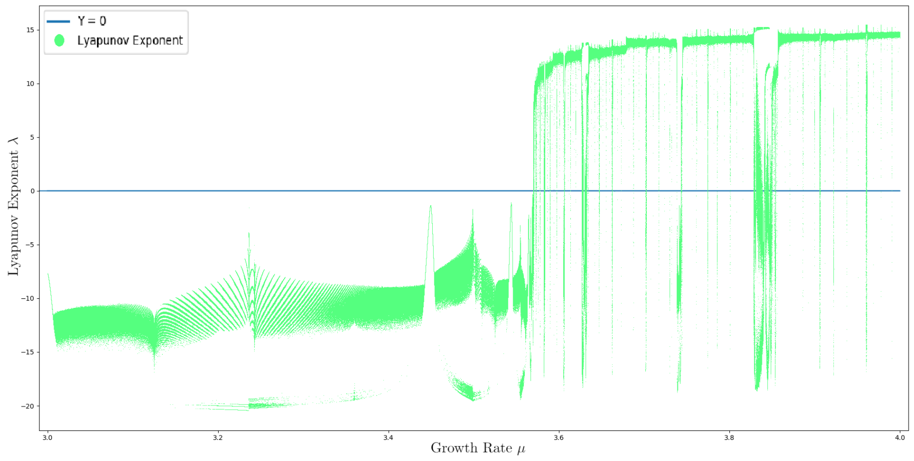

Figure 2.7: Lyapunov Exponent λ as a function of the Growth Rate μ for the logistic map$f^{n+1}(x) = \mu f^n(x) \times (1 - f^n(x))$, where $\mu \in [3, 4]$. **Positive** λ values signify sensitivity to initial conditions.

As discussed in 2.3.2 **Methodology and the Code**, the Lyapunov Exponent has been solved for the logistic map

$$f^{n+1}(x) = \mu f^n(x) \times (1 - f^n(x)) \tag{2.14}$$

The key variable in this equation is μ. It is also referred to as the Growth Rate element, as it controls how $f^{n+1}(x)$ grows in comparison to $f^n(x)$.
For $0 < \mu < 1$, $\quad f^{n+1}(x) \ \rightarrow \ 0$ as $n \rightarrow \infty$,
for $1 < \mu < 3$, $\quad f^{n+1}(x) \ \rightarrow$ const. as $n \rightarrow \infty$,
for $3 < \mu < 3.56$, $\quad f^{n+1}(x)$ oscillates between several values
for $3.56 < \mu < 4$, $\quad f^{n+1}(x)$ becomes chaotic.

Fig.2.7 shows the Lyapunov exponent for $\mu \in [3, 4]$ as that is where the system behaves dynamically (does not settle on one value over time) and is of interest to us. The first thing one notices is that even where the system is dynamical it is not necessarily chaotic. The reason for this is reviewed in detail in 3 **Towards an Understanding of Chaos**. λ becomes positive for the first time at **3.56986** as seen in fig.2.8.

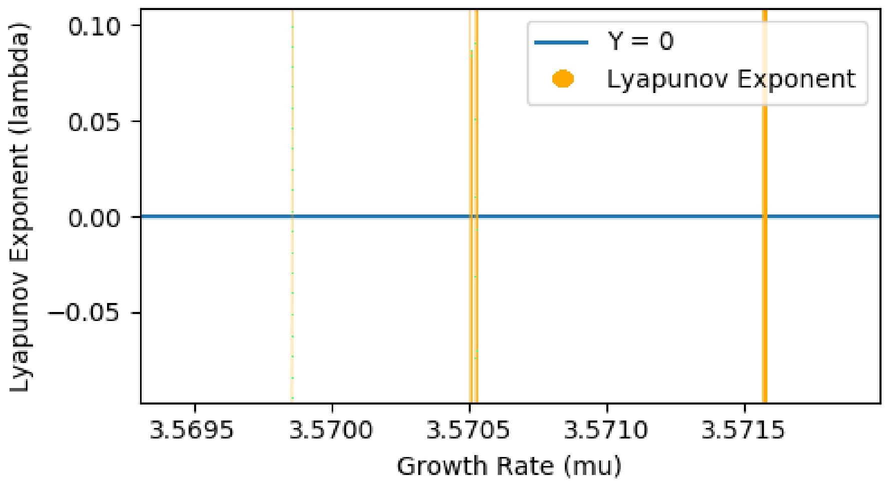

Figure 2.8: Close up of fig.2.7. Lyapunov Exponent λ becomes positive at $\mu = 3.56986$

Obtaining the uncertainty in this value is not a straightforward process. Traditionally uncertainty would be obtained as:

$$\lambda = \frac{1}{n}\sum_{i=0}^{99} ln\left|\frac{g^n(x_i) - f^n(x_i)}{\Delta x}\right| \tag{2.15}$$

Where, like in 2.3.2 **Methodology and the Code**, $f^n(x_i + \Delta x) = g^n(x)$.

$$d\lambda = \sqrt{\left(\frac{d\lambda}{dg}\Delta g\right)^2 + \left(\frac{d\lambda}{df}\Delta f\right)^2} = \frac{1}{100}\sum_{i=0}^{99}\frac{1}{g^n(x_i) - f^n(x_i)} \tag{2.16}$$

In this case Δg and Δf would be the limits of accuracy touched on in 2.3.2, which are 26 digits of accuracy. If one were to solve the above equation he would obtain $d\lambda \approx 10^{-21}$. Unfortunately this is not the case. Error arises long before the program runs out of digits to store data in. Manipulating Δx (in $f^n(x + \Delta x)$) results in significant changes, as seen in fig. 2.9.

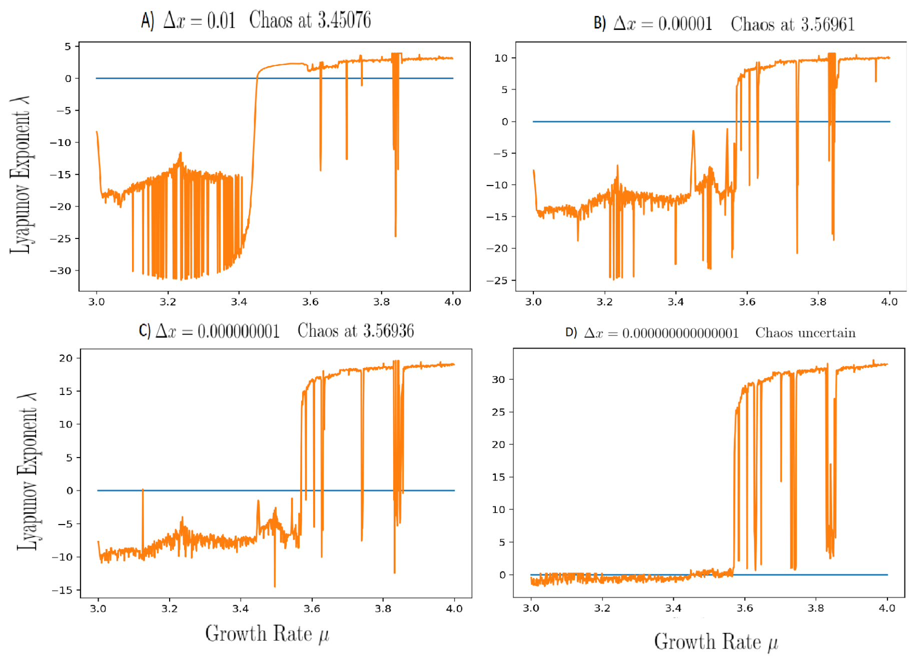

Figure 2.9: Variations of fig.2.7 with different values for Δx. **A)**$\Delta x = 10^{-3}$, Chaos onset at $\mu = 3.45076$; **B)**$\Delta x = 10^{-6}$, Chaos onset at $\mu = 3.56961$; **C)**$\Delta x = 10^{-10}$, Chaos onset at $\mu = 3.56936$; **D)**$\Delta x = 10^{-16}$, Chaos onset uncertain;

Intuition tells us that sending $\Delta x \to 0$ should result in more precise results. In reality there is a restriction on precision imposed by the program used to generate these results. As discussed in 2.3.2 a limitation of 26 digits of accuracy is imposed. For very small values of Δx and non-chaotic values of μ ($\mu \in [0,\ 3.56]$), the difference $g^n(x) - f^n(x)$ becomes too small to be expressed within 26 digits, effectively running for too few iterations to allow for a good estimate of λ. The author believes this can be overcome by some clever coding which solves the underlying problem of

$$ln \left| \frac{g^n(x) - f^n(x)}{\Delta x} \right| = ?\ \ ;\ g^n(x) - f^n(x) = 0 \tag{2.17}$$

What does this mean for our uncertainty in λ? It is difficult to estimate the exact effect all factors have on the value of the Lyapunov Exponent. If we select a value for Δx that is sufficiently small yet large enough to allow for a satisfactory number of iterations, such as the in fig.2.7, the value of μ at which chaos sets in is at

$$3.56986 \pm 0.0005 \tag{2.18}$$

,which agrees with results obtained mathematically (Chaos onset at $\mu = 3.569945672$) [Weia]. Thus it can be concluded with 99.9%c.l for values of $\mu \geq 3.7$ the map $f^{n+1}(x) = \mu f^n(x) \times (1 - f^n(x))$ has sensitive dependence on initial conditions, and thus satisfies the third and final definition of a chaotic system.

Chapter 3

Towards an Understanding of Chaos

In this Chapter we will go towards a deeper understanding of the map f and the deterministic chaos it gives rise to.

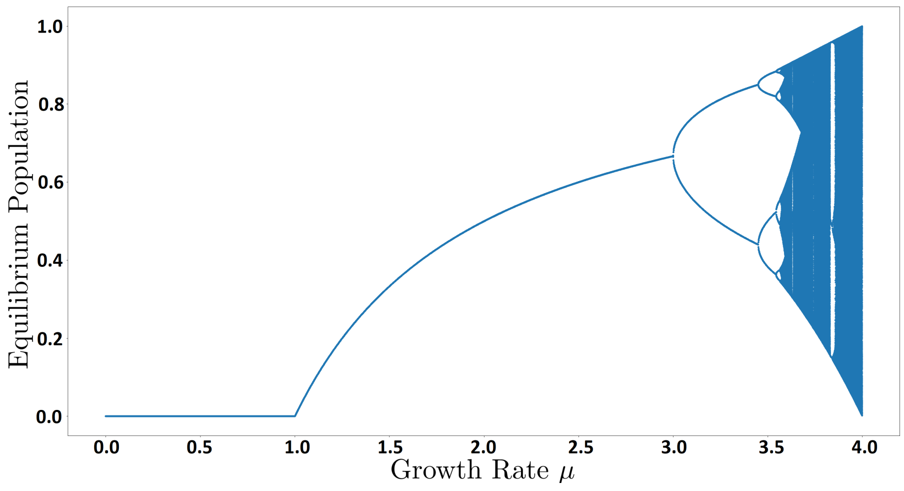

Figure 3.1: The Logistic Map Sec. A.3. The values f converges to, against Growth Rate μ. Equilibrium achieved with 20'000 iterations

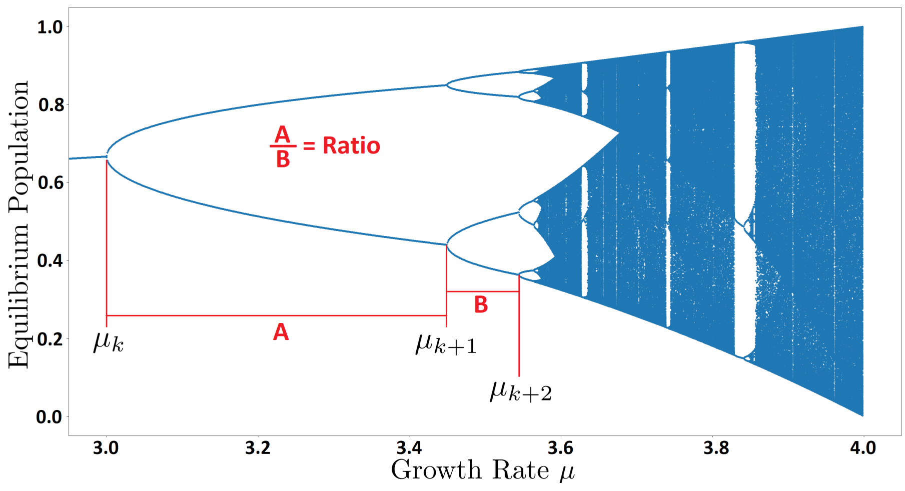

Figure 3.2: Close up of the Logistic Map Fig.(3.1) and illustration of Eq.(3.1)

When discussing a 1D map such as f, the Logistic Map (Fig.3.1) presents the systems behaviour in the most intuitive way. At first converging to 0 and then to a positive number. At $\mu = 3$, the first bifurcation occurs - the system now oscillates between two stable states. As μ grows the number of those states becomes 4, 8, 16, 32, 64, 128, 256, ..., until at $\mu \approx 3.570$, f no longer converges and becomes transitive. It is not unreasonable to ask: *"Is the chaos we observe just not the system having an very large number of stable states?"*. The answer to that is a definite **No**. The difference is that a system with a large amount of stable states will only oscillate around those states, while a transitive system will generate any value in the co-domain of the map given enough iterations. What this means for us is that for a real life system described by a chaotic map, we cannot guess at its state out of the hypothetical 2^n states available, as every state is a possible state and thus guessing becomes meaningless.

3.1 Feigenbaum's number

There is a peculiar order leading up to the point of chaos ($\mu \approx 3.570$). Let us define the number of bifurcations with k, so that at $k = 1$, the number of 'branches' or **Period** $= 2$.
Thus at $k = (2,\ 3,\ 4,\ k)$ the Period $= (4,\ 8,\ 16,\ 2^k)$. If we were to take the length of a branch (A) and divide by the length of the branch that begins from our initial one (B) we would observe that there is a ratio which hold regardless of which pair we choose. That is to say, the system is asymptotically periodic. This is illustrated in fig.3.2.

$$Ratio\ \ \delta_k \ = \ \frac{\mu_{k+1} \ - \ \mu_k}{\mu_{k+2} \ - \ \mu_{k+1}} \tag{3.1}$$

This ratio is known as Feigenbaums constant and is represented by δ. There are several different versions of δ, depending on the dimension of the map, however in this work we will focus solely on the 1D case. Feigenbaum also has a constant signified by α which has to do with the width of the branches. In his paper from 1978 for the Journal of Statistical Physics [Fei78] he obtained $\delta = 4.669201609103....$ This number was later improved upon to 84 places by Briggs (1991) [Bri91] and again to 576 places in (1997) [Bri97]. Two years later Broadhurst, in an email to his friend, defined it to 1018 places [Bro99].

Obtaining a value of our own will be the subject of the few following sections. There are several methods one can approach this. One being simply plotting the Logistic Map or generating a list of the values and measuring the distance between bifurcations. This was the method utilised by the author in the early stages of this project, alas it is severely limited in precision due to the constraints of computer arithmetic. The best value obtained, $\delta \ = \ 4.920$ is only a vague suggestion towards the true ratio.
Another method, the one utilised by Feigenbaum [Fei78], is using power series approximations. Below we will explore in detail a variety of different approaches to this problem and compare results. Should the reader wish to cut to the most accurate method, skip to the Direct Method as devised by Briggs in 1989 [Bri89].

3.1.1 Looking at the Figure

This is perhaps the least sophisticated approach. We take Fig.3.1 and look at it really hard. Alas as we magnify the bifurcation points to pinpoint exactly where they begin, a complication arises. As seen in Fig.(2.2) we cannot correctly determine where the branch begins due to two limitations imposed by the capabilities of the machine and time. The first being the impossibility to fully converge each system. A system is f^n iterating continuously at a specific Growth Rate μ. The system will converge at $n \to \infty$ which is not obtainable. The second limitation is the limitation on sample size. A sample is the number of different values of μ for which f is iterated. If we desire a large sample size in order for points to be denser, we limit ourselves in terms of the number of iterations n we have time to compute. As each system must go through the same number of iterations as the rest, increasing the systems quantity lengthens computation time.

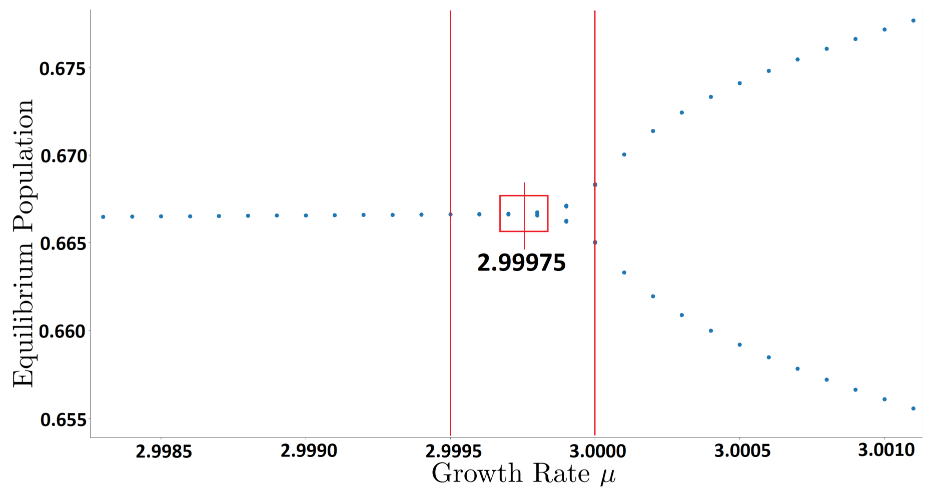

Figure 3.3: Close up of Fig.(3.1), vertical space between the points is due to the system not fully converging, horizontally the data is distanced due to the finite number of samples in the region $\mu \in [0,\ 4]$.

Regardless with an attempt to reasonably balance sample size and iteration length we plot Fig.(3.1) in detail and to the best of our abilities the following data is extracted:

k	Branches $= 2^k$	μ_k	Difference $\mu_k - \mu_{k-1}$	Ratio δ_k
0	1	0	-	-
1	2	2.99975	2.99975	-
2	4	3.44938	0.44963	6.671597
3	8	3.54076	0.09138	**4.920442**
4	16	3.56439	0.02363	3.867118
5	32	3.56876	0.00437	5.407323

Table 3.1: The ratio δ_k as estimated from data gathered by measuring the distance between points in Fig.(3.1). Best estimation is 4.920 which hint towards the true value of 4.669.

While it is possible to overcome some of these obstacles by only generating samples over the areas of interest, the true ratio δ is achieved by computing the ratio between bifurcation points separated by infinitesimally small distances; Eq.(3.2). The above statement implies that there is a very harsh limit to the achievable precision of this method.

$$\delta = \lim_{k\to\infty} \delta_k \tag{3.2}$$

3.1.2 The Fixed Point Analysis Method

This method resolves the issue of limited precision the previous approach has by mathematically obtaining the exact position at which the bifurcation occurs instead of converging a system towards it. Fixed points reside within a single state, not changing upon application of a map [Weib]; Eq.(3.3).

$$f^n(x) = x \tag{3.3}$$

The first stable point in the Logistic Map can be determined by Eq.(3.4).

$$f^1(x_0) = x_0 \tag{3.4}$$

$$\mu\, x_0\,(1 - x_0) = x_0 \tag{3.5}$$

$$\mu\, x_0\,(1 - x_0) - x_0 = 0 \tag{3.6}$$

$$\mu\, x_0 \left(x_0 - \left(\frac{1-\mu}{\mu} \right) \right) = 0 \tag{3.7}$$

The solutions to this equation, $x_0 = 0$ and $x_0 = (1-\mu)/\mu$ are the 1-st order Fixed points. The values of μ at which the stable points occur can be computed by solving the discriminant for μ. The equations below were evaluated using Wolfram Mathematica v12.0.

$$Discriminant[x_0,\ \mu] \tag{3.8}$$

$$Discriminant[0,\ \mu] = 0 \tag{3.9}$$

$$Discriminant[(\mu-1)/\mu,\ \mu] = 1 \tag{3.10}$$

Our 1-st order Fixed Points occur at $\mu = 0,\ 1$. At this order no bifurcations have developed. To determine the point at which the Logistic Map experiences its first doubling we would have to solve $f^2(x_0) = x_0$.

$$\mu\, x_1\,(1 - x_1) = x_0 \tag{3.11}$$

$$\mu\,(\mu\, x_0\,(1-x_0))\,(1 - (\mu\, x_0\,(1-x_0))) = x_0 \tag{3.12}$$

$$\mu\,(\mu\, x_0\,(1-x_0))\,(1 - (\mu\, x_0\,(1-x_0))) - x_0 = 0 \tag{3.13}$$

There are four solution's to this equation:

$$x_0 = 0 \tag{3.14}$$

$$x_0 = (1-\mu)/\mu \tag{3.15}$$

$$x_0 = \frac{\mu + \mu^2 \pm \mu\sqrt{-3 - 2\mu + \mu^2}}{2\mu^2} \tag{3.16}$$

As the 2-nd order system is an iteration of the 1-st, $f(f^1(x_0)) = x_0$, its solutions include those for the 1-st order system. The last two solutions will only be real if the root is 0 or positive, therefore we can simply solve the discriminant of what is inside it:

$$Discriminant[-3 - 2\mu + \mu^2,\ \mu] \ = \ 3 \tag{3.17}$$

$\mu = 3$ is therefore the point where the 1-st period doubling originates. From this point on equations rapidly increase in size and the author will display only enough to prove his point. We will not be solving the 3-rd order system as the solutions to that one belong to a map of higher dimension. An oversimplified way of figuring out which order systems are of interest is that a 2-nd order system reveals where the Logistic Map bifurcates into a total of 2 branches. A 4-th order system reveals the start of 4 branches, 8-th order: of 8, etc. Without further delay we attempt to find our next period doubling point: $f^4(x_0) = x_0$.

$$\mu\, x_3\,(1 - x_3) \ = \ x_0 \tag{3.18}$$

$$(\mu(1-x_0)x_0)((\mu(1-x_0)x_0)((\mu(1-x_0)x_0)((\mu(1-x_0)x_0)(x_0)))) - x_0 \ = \ 0 \tag{3.19}$$

There are 16 solutions to this, four of which are the solutions for the 1-st and 2-nd order system. The rest are roots to different powers, the contents of which are identical.

$$x_0 \ = \sqrt{(\ 1 + \mu^2 + ... + (-\mu^9 - 15\mu^1 1 - 20\mu^1 2)x_0^9 + (3\mu^1 1 + 15\mu^1 2)x_0^1 0 - 6\mu^1 2x_0^1 1 + \mu^1 2x_0^1 2)} \tag{3.20}$$

This solution is not as straightforward as the previous ones as it is dependent on x_0. Setting whats inside the square root equal to 0 and solving once again for x_0 however gives us the desired expressing in terms of μ:

$$x_0 \ = \ 1037970703125\mu^1 32 + ... + \mu^1 72 \tag{3.21}$$

At this point we have nearly made it. We divide all 36 terms by μ^{132} which is the one with to the smallest power.

$$\frac{(-5 - 2\mu + \mu^2)^2(5 - 4\mu + 6\mu^2 - 4\mu^3 + \mu^4)^3(-135 - 54\mu - 9\mu^2 + 28\mu^3 + 3\mu^4 - 6\mu^5 + \mu^6)^4}{\mu^{132}} \tag{3.22}$$

We take the discriminant of the numerator as that is what interests us:

$$Discriminant[(-5 - 2\mu + ... + \mu^6)^4,\ \mu] \ = \ 1 + \sqrt{6} \ = \ 3.4495, \tag{3.23}$$

There are multiple other solutions to the Discriminant, however most are not of interest to us as they are either imaginary, or negative. There are two legitimate solutions, the one above and $\mu = 3.9601$ which is the start of 4 branches at the end of one of the islands of non-chaotic behaviour as seen in Fig.(3.2).

Unfortunately we have hit yet another roadblock. The next Fixed point at $f^8(x_0) = x_0$ has 256 (2^8) solutions for x_0 meaning that many of the polynomials will be above 200-th order. The author believes computation time for the above to range between several hours to a few days depending on the machine its being solved on. One thing is certain and it is that this method is no longer viable.

Order of fixed point	k	Branches = 2^k	μ_k	Difference $\mu_k - \mu_{k-1}$	Ratio δ_k
1	0	1	0	-	-
2	1	2	3	3	-
4	2	4	3.44949	0.44994	6.674

Table 3.2: The ratio δ_k as estimated from data gathered by computing fixed points

While we have figured out the points to considerable accuracy, they are so early in the series that the only ratio $\delta = 6.674$ we are able to obtain is far from the true value.

3.1.3 The Direct Method

In the family of functions $f:\ X \to X$, so far we have been looking strictly at the map $f^{n+1}(x_n) = \mu x_n(1 - x_n)$. There are however maps with equivalent properties and to make things easier for ourselves we will be using one of them for this method rather than the one studied up to this point. The reason this map is simpler is that its derivative $\partial/\partial\mu$, has an independent constant allowing us to set variables to 0.
The map in question is:

$$h^{n+1}(x_n) = \mu - x_n^2 \tag{3.24}$$

Similarly to how we proved our map is dense in X in Par.**2.2** we will utilise those techniques to study the parameters of the new system.The domain of X is

$$\{x \in \mathbb{R} \mid -2 \leq x \leq 2\} \tag{3.25}$$

Max($h^{n+1}(x_n)$) lies at $x_n = 0$

$$\frac{h^{n+1}(x_n)}{x_n} = -2x_n = 0 \tag{3.26}$$

From Eq.(3.27) we conclude that the range of the Growth Rate is $\mu \in [0,\ 2]$

$$Max(h^{n+1}(x_n)) = 2 = \mu - 0^2 = \mu \tag{3.27}$$

We now push the following notion: Let μ_k^i be a point at which the i^{th} period doubling has occurred: the system has stabilised into 2^i branches. Further let us define a super-stable point μ^i as the exact value at which the 2^i branches occur. We can think of the points we obtained in Sec. (**3.1.2**) as super-stable. Furthermore we define Feigenbaums constants true value as Eq.(3.28).

$$\delta = \lim_{i\to\infty} \left(\frac{\mu^{i-1} - \mu^{i-2}}{\mu^i - \mu^{i-1}} = \delta^i \right) \tag{3.28}$$

$$\delta^i = \lim_{k\to\infty} \left(\frac{\mu_{k-1}^i - \mu_{k-2}^i}{\mu_k^i - \mu_{k-2}^i} = \delta_k \right) \tag{3.29}$$

What we are doing is we are taking some points μ_k^i at which we are certain a period doubling has already occurred then we iterate polynomials which are stable at 0, similarly to our approach in the previous section. The difference being that in our previous approach we stabilised the polynomial by equating to 0 and attempted to obtain the exact position with "one" calculation. While with this method we approach μ^i from some stable point between μ^i and μ^{i+1}; [Bri89].

Polynomials are iterated similarly to h, however they are a function of μ.

$$p_k(\mu) = \mu - [p_{k-1}(\mu)]^2 \tag{3.30}$$

$$p_0(\mu) = 0 \tag{3.31}$$

Iff $p_k(\mu) = 0$, h_μ has a super-stable point from which 2^k branches originate.
Rearranging Eq.(3.29) we obtain an equation for μ_k:

$$\mu^i = \mu^{i-1} + \frac{\mu^{i-1} - \mu^{i-2}}{\delta^i} \tag{3.32}$$

$$\mu_{k+1}^i = \mu_k^i + \frac{p_k(\mu_k^i)}{p_k'(\mu_k^i)} \tag{3.33}$$

$$p_k'(\mu_k^i) = \frac{\partial\left(p_k(\mu_k^i)\right)}{\partial\mu} = 1 - 2\,p_{k-1}'\,p_{k-1} \tag{3.34}$$

$$\mu^i = \lim_{k\to\infty} \mu_k^i \tag{3.35}$$

Applying the above with Eq.(3.28) allows us to obtain Feigenbaums constant with increasing accuracy. Eq.(3.32) estimates the position of the i^{th} super-stable point. Then the value of that point is refined by iterating Eq.(3.33), where the derivative of the polynomial with respect to μ is defined in Eq.(3.34). Before we attempt to estimate the following super-stable point, a new improved estimate for δ is obtained through Eq.(3.28), allowing the next super-stable point to be obtained to greater accuracy. The whole process seems to increase precision linearly with every two iteration producing another true digit [Cod].

The Code

```
1  mu1 = 1.0             # This is Growth rate mu^(i-1)
2  mu2 = 0.0             # mu^(i-2)
3  f = 4.92             # Best approx. of Feigenbaums constant
4
5  for i in range(2, max_it + 1):
6      mu = mu1 + (mu1 - mu2) / f    # mu is mu^(i)
7      for k in range(1, max_it_k + 1):
8          p = 0.0
9          dp = 0.0
10         for n in range(1, (1 << i) + 1):
11             dp = 1.0 - 2.0 * dp * p
12             p = mu - p**2
13         mu = mu - p / dp
14     f = (mu1 - mu2) / (mu - mu1)
15
16     mu2 = mu1
17     mu1 = muue.
```

Listing 3.1: Code approximating Feigenbaums constant

The Code utilizes the method developed by Briggs [Bri89], and its core is inspired by an anonymous contributor at RosetaCode.org [Cod].

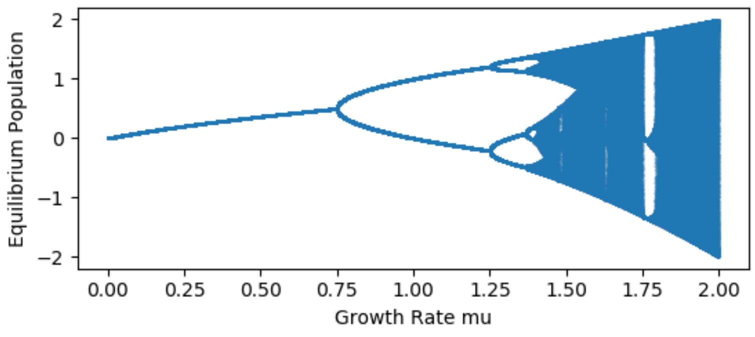

Figure 3.4: Branch Plot of the continuous map $f^{n+1}(x_n) = \mu - x_n^2$ belonging to the family $f: X \to X$

On lines 1 and 2 we define the first two bifurcation points: $\mu^{i-2=0} = 0$ and $\mu^{i-1=1} = 1$. $\mu^0 = 0$ is trivial, however notice that $\mu^1 = 1$ is a rough estimate. From Fig.(3.4) we see that the first period doubling beings at $\mu^1 \approx 0.75$ region, however we want to choose a stable point between μ^1 and μ^2, thus we set $\mu^1 = 1$.
As we enter the first `for` cycle, we solve Eq.(3.32) on line 6. Using this newly obtained estimate for μ^3 we iterate Eq.(3.33) which results in μ^3 converging towards its true value as per Eq.(3.35). Now that we have obtained a good estimate for our first three values for μ we reevaluate the Feigenbaum constant on line 14. Before we begin the process anew all values for mu get shifted to the right: $\mu^{i-2} = \mu^{i-1}$, $\mu^{i-1} = \mu^i$. This method is mostly limited by the precision of the calculations. Without additional improvements to the code, meaning 16 digits of accuracy with Python, one begins diverging from the Feigenbaum constant after roughly 15 `i` iterations. This is of course dependant on the number of iterations we choose to complete for `k`, however it itself requires digits of precision. Overall there are many ways one can manipulate the code to obtain better results, if one is ready to wait during the increased computation time. Sec. A.8

Feigenbaums Constant - The Closest Estimate

The closest estimate we were able to obtain was with our final method: $\delta = 4.6692016$, Table (3.3). This result has 8 digits of accuracy, which includes numbers before the decimal separator. For practical purposes this is a satisfactory result, nevertheless there is allure in searching for higher and higher levels of precision. The author advises one on such a quest to further explore the Direct Method as well as read the e-mail by Broadhurst [Bro99] as in it he outlines the methodology used to obtain his value, considered to be the best estimate to present day with 1018 digits.

i	δ	Correct Digits
1	4.92000000	1
2	3.21851142	0
3	4.38567760	1
4	4.60094928	2
5	4.65513050	2
6	4.66611195	3
7	4.66854858	4
8	4.66906066	4
9	4.66917155	5
10	4.66919515	6
11	4.66920019	6
12	**4.66920169**	**8**
13	4.66920465	6
14	4.66919371	6
15	4.66926610	5
20	4.66847161	0
True Value	4.669201609	-

Table 3.3: Feigenbaums constant as computed by the Direct Method. Closest estimate obtained is $\delta = 4.66920169$.

3.2 Cobweb

In the next few sections we will be looking to increase are intuition with chaotic systems of this type. The next Figure (Fig. (3.5)) shows a plot with three lines. The blue line is a curve described by some map f. The map used for Fig. (3.5) is the familiar $f^{n+1}(x_n) = \mu x_n(1 - x_n)$. Notice the shape recognisable as that of a quadratic function. Feigenbaum has shown that all one dimensional maps with a single quadratic maximum bifurcate with a period $\delta = 4.669$. [Fei78]. Naturally both maps that were explored in this work fit that description. The second, amber line is simply an $y = x$ line. The green line appears to be "bouncing" off the previous two. Each line serves as a connection between two points. Disregarding the very first point, which serves to give the system its initial conditions, each point lies on either the blue or amber line, alternating with each successive point as described in Eq.(3.36). Sec. A.4

$$[f(x),\ x] \to [f(x),\ f(x)] \to [f^2(x),\ f(x)] \to [f^2(x),\ f^2(x)] \to [f^3(x),\ f^2(x)] \to etc... \tag{3.36}$$

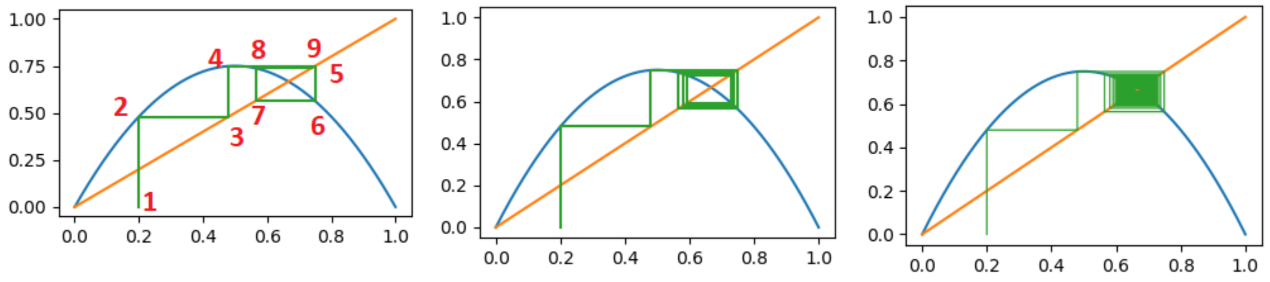

Figure 3.5: Cobweb plot of the map $f^{n+1}(x_n) = \mu x_n(1-x_n)$; $\mu = 3$, $x_0 = 0.2$. On the leftmost figure, successive points are numbered. Both axes represent some value obtainable by $f(x)$ and their range is the range of the map: $[0,\ 1]$.

On the leftmost figure in Fig. (3.5) successive points are numbered to highlight the pattern. On its right there are figure which illustrate the continuation of the pattern. What we notice is that the points begin to converge. Cobweb plots are very useful when it comes to studying the areas where a map converges under certain conditions. In the above example, it converges to a single point, although that is not at all necessary. It can converge on two, four, eight or any other number (powers of 2 only for this specific map) of values. Below in Fig. (3.6) a Cobweb converging on four points can be observed. It has been let stabilised over 1500 iterations before any values have been recorded thus making it easier for the reader to see clearly the values of convergence.
The really interesting thing right now is **How does a cobweb look for a chaotic system?**. Fig. (3.7) shows precisely that. On first glance it seems to be exactly what one might expect. From the 32 iterations recorded (after stabilising), 32 produce a unique point. If we keep on plotting points however, it becomes obvious there is an order within the chaos. The system follows a specific path and it follows it, each time with a small translation. An

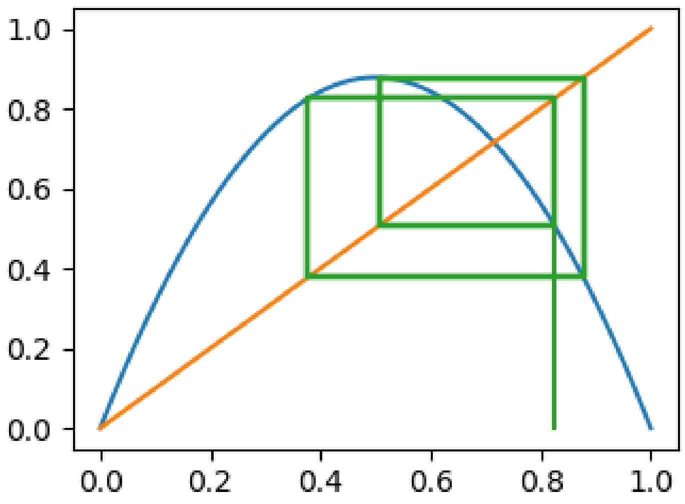

Figure 3.6: Cobweb plot of the map $f^{n+1}(x_n) = \mu x_n(1 - x_n)$; $\mu = 3.5$, $x_0 = 0.2$. The system has undergone 1500 iterations before any values being recorded, and is therefore stabilised over 4 points. The reader might find it easier to decipher which those points are by looking solely at the $y = x$ line.

oversimplified analogy would be if I were to count $1, 2, 3, 4, 5 \rightarrow 1.1, 2.1, 3.1, 4.1, 5.1 \rightarrow 1.2, 2.2, 3.2, 4.2, 5.2$ etc. As the system nears its "end" where it would begin converging Fig. (3.7) it always escapes renewing the cycle, this time a little bit differently than the last.

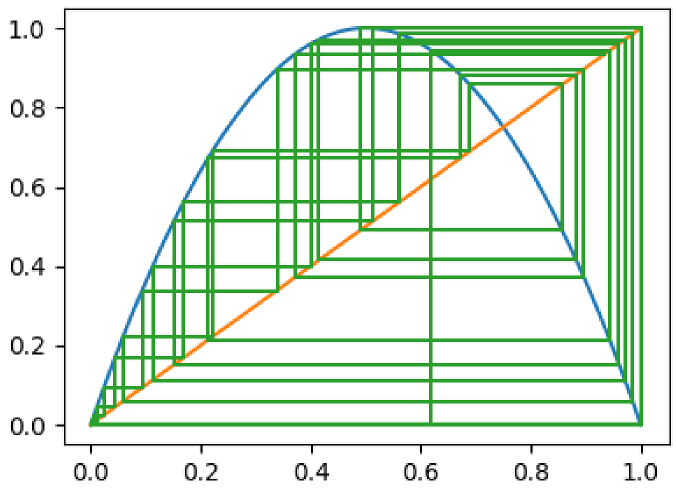

Figure 3.7: Cobweb plot of the map $f^{n+1}(x_n) = \mu x_n(1 - x_n)$; $\mu = 4$, $x_0 = 0.2$. The system has been "stabilised" over 1500 iterations. From 32 recorded data points, 32 are unique.

3.3 Fourier Transform of Time Series

First a Time Series of a system is generated, stabilised over 10'000 iterations and then 1000 data points are collected. Here a system is defined to be $f(x)$ for some specific Growth Rate μ. A time series for a non chaotic should should oscillate between some set number 2^k fixed points. Then a fast Fourier Transform is taken of the series, with the results displayed in Fig. (3.9). Sec. A.5
In the non-chaotic regions the Fourier transform follows the pattern $2^k - 1$, where k is the number of consecutive period doublings as defined in previous sections. It appears that doing the Fourier Transform of a chaotic system's time series returns random noise. In an abstract setting as the one studied the Fourier transform does not seem to tell us anything beyond the 2 facts stated above. The reader is advised to perform further research before deciding on the usefulness of this method.

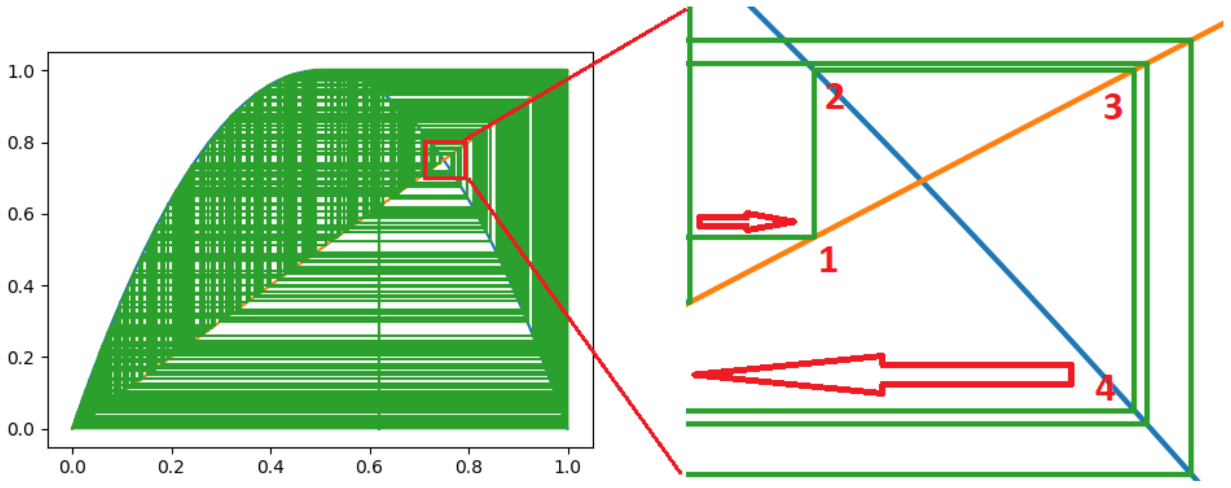

Figure 3.8: Left: Continuation of Fig. (3.7). Right: close up of the centre where the system attempts converging.

3.4 Universality and Applications

In this work we have encountered two maps: $f^{n+1}(x_n) = \mu x_n(1-x_n)$ and $f^{n+1}(x_n) = \mu - x_n^2$. However bifurcation maps can be observed for any one-dimensional maps with a single quadratic maximum. A comprehensive list can be found at [Wik]. Feigenbaums constant is believed to be transcendental and it is yet uncertain whether it is a fundamental constant like π and e. [Kni].
Deterministic chaos manifests itself in many forms. Although this is a Physics paper, it would be injustice to the topic to discuss it strictly as one type of phenomenon. The more striking instances of deterministic chaos which the author encountered during his study were "Synchronous period-doubling in flicker vision of salamander and man" [DW98], "Dynamics of period-doubling bifurcation to chaos in the spontaneous neural firing patterns" [Jia12], "Controlling cardiac chaos"[Gar92], "Chaos and the Dynamics of Biological Populations"[May87], "Low dimensional chaos in cardiac tissue"[D90], "A new modified resource budget model for nonlinear dynamics in citrus production"[Ye16] and "Period doubling cascade in mercury, a quantitative measurement"[A82].

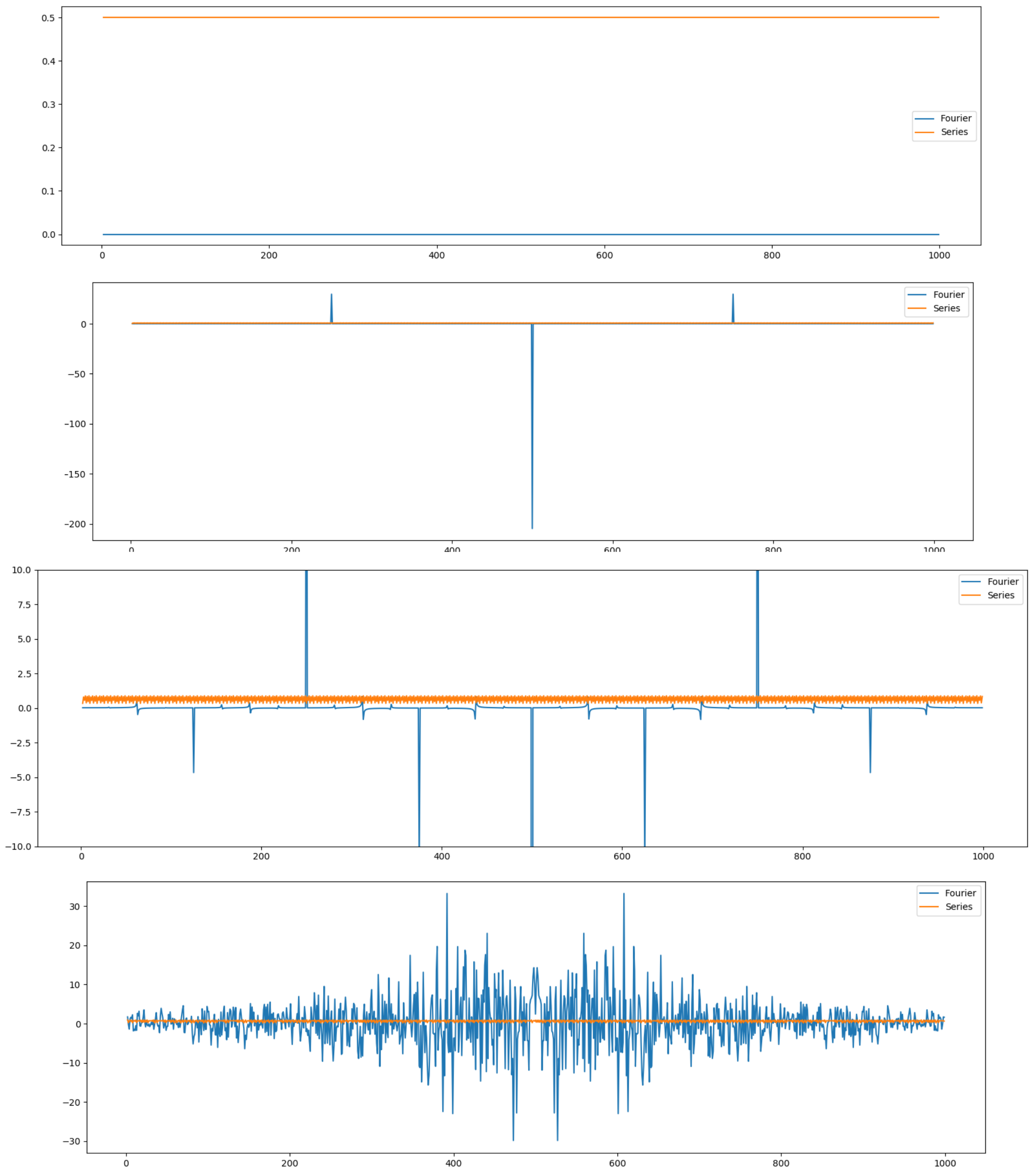

Figure 3.9: Time Series and its Fast Fourier Transform. From top to bottom: 1)$\mu = 2$, Stable points = 1, Peaks = 0; 2)$\mu = 3.2$, Stable points = 2, Peaks = 1; 3)$\mu = 3.569$, Stable points = 32, Peaks = 31; 1)$\mu = 3.8$, Stable points = none, Peaks = ?;

Bibliography

[Gar92] A Garfinkel. “Controlling cardiac chaos”. In: *Science* Vol. 257.Issue 5074 (28 Aug 1992), pp. 1230–1235.

[May87] R. M. May. “Chaos and the Dynamics of Biological Populations”. In: *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences* Vol. 413.No. 1844 ((Sep. 8, 1987)), pp. 27–44.

[Fei78] Mitchell J. Feigenbaum. “Quantitative Universality for a Class of Nonlinear Transformations”. In: *Journal of Statistical Physics* 19.1 (1978), pp. 25–52.

[A82] Libchaber A. “Period doubling cascade in mercury, a quantitative measurement”. In: http://dx.doi.org/10.1051/jphy 43. 10.1051/jphyslet:01982004307021100 (1982).

[Bri89] K. Briggs. “How to Calculate the Feigenbaum Constants on Your PC.” In: *Austral. Math. Soc.* 16.1 (1989), pp. 89–92.

[Dev89] R.L. Devaney. *An Introduction to Chaotic Dynamical Systems.* Addison-Wesley, 1989.

[D90] Chialvo D. “Low dimensional chaos in cardiac tissue. Nature”. In: *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences* https://doi.org/10.1038/343653a0 (1990).

[Bri91] K. Briggs. “A Precise Calculation of the Feigenbaum Constants.” In: *Math. Comp.* 57.1 (1991), pp. 435–439.

[Bak96] G.L. Baker. *Chaotic Dynamics, an Introduction.* Second Edition. Cambridge University Press, 1996.

[Bri97] K. Briggs. “Feigenbaum Scaling in Discrete Dynamical Systems.” In: *Ph.D. thesis. Melbourne, Australia: University of Melbourne* (1997).

[DW98] Crevier DW. “Synchronous period-doubling in flicker vision of salamander and man”. In: *Neurophysiol.* Apr;79.4 (1998), pp. 1869–78.

[Bro99] Broadhurst. “Feigenbaum Constants to 1018 Decimal Places.” In: *Email dated 22-Mar-1999. https://pi.lacim.uqam.c* (1999).

[Ban10] J. Banks. “On Devaney’s Definition of Chaos”. In: *The American Mathematical Monthly* 99.4 (2010), pp. 332–334.

[Gro11] Karl-G. Grosse-Erdmann. *Linear Chaos.* Springer, 2011.

[Jia12] Bing Jia. “Dynamics of period-doubling bifurcation to chaos in the spontaneous neural firing patterns Cogn Neurodyn (2012)”. In: DOI 10.1007/s11571-011-9184-7 (2012).

[Cra15] A. Crannell. “The Role of Transitivity in Devaney’s Definition of Chaos”. In: *The American Mathematical Monthly* 102.9 (2015), pp. 788–793.

[Ye16] Xujun Ye. “A new modified resource budget model for nonlinear dynamics in citrus production”. In: *Nature* Solutions and Fractals.87 (2016), pp. 51–60.

[Cod] Rosetta Code. *Feigenbaum Constant Calculation.* URL: `https://rosettacode.org/wiki/Talk:Feigenbaum_constant_calculation`. (accessed: 22/04/2020).

[Cor] Carlos Cordoba. *No multiprocessing Print output.* URL: `https://stackoverflow.com/questions/48078722/no-multiprocessing-print-outputs-spyder`. (accessed: 12/04/2020).

[Kni] Oliver Knill. *Mitchell Feigenbaum (1944-2019), 4.66920160910299067185320382...* URL: `https://writings.stephenwolfram.com/2019/07/mitchell-feigenbaum-1944-2019-4-66920160910299067185320382/` (accessed: 27/04/2020).

[Weia] Eric W. Weisstein. *Logistic Map.* URL: `https://mathworld.wolfram.com/LogisticMap.html`. (accessed: 14/04/2020).

[Weib] Eric W. Weisstein. *Logistic Map.* URL: `https://mathworld.wolfram.com/FixedPoint.html`. (accessed: 22/04/2020).

[Wik] Wikipedia. *List of Chaotic Maps.* URL: `https://en.wikipedia.org/wiki/List_of_chaotic_maps`. (accessed: 27/04/2020).

Appendices

Appendix A

The Code

A.1 Running the Code

If this section is to be of any use to those interested in it, some non-trivial background information has to be given if the reader is to replicate the code.
The machine used to run the code has the following properties:
OS: Windows 10
IDE: Spyder 4.1.2
CPU: AMD Ryzen 5 2600 Six-Core Processor (2 logical cores per physical)
The significance of the OS is the way multiprocessing is handled by the Python IDLE. With Windows the children of the main process are not allowed to interact with the IDLE, thus if one wished to run the code this has to be executed in an external terminal [Cor]. This can be done on Spyder by
Tools→Preferences→Run→Console:Execute in an external system terminal.

The significance of the CPU type comes from the fact the author has utilised multiprocessing to solve the problem of long execution times that comes with generating large amounts of data (>1'000'000'000 calculations required). The software has been written to make use of 10 logical cores, and execution takes 79 minutes on average.

A.2 Lyapunov Exponent

```python
  # -*- coding: utf-8 -*-
"""
Created on Fri Apr 10 22:01:04 2020

@author: 967869@swansea.ac.uk
"""
#Imports -----------------------------------------------------------------------
import decimal as d
import multiprocessing as mp
import time
import matplotlib.pyplot as plt
import numpy as np

#Define constants ------------------------------------------------------------
START_TIME = time.time()
N = 10000
DX = 0.00000001
RESOLUTION = 1000

#   f(x) -----------------------------------------------------------------------
def lyapunov(rate):
    """Returns Lyapunov exponent for rate over N gen."""
    function = 0.2
    d_function = function+DX
    sigma = 0

    for i in range(N):
        function = rate * function * (1 - function)
        d_function = rate * d_function * (1-d_function)

        if d_function-function == 0:
            break

        sigma += d.Decimal(abs(d_function - function) / DX).ln()

    sigma = sigma / N #(i + 2)
    return sigma

#Thread  functions--------------------------------------------------------------
def loop(start, stop, X, Y):
    """Runs Lyapunov over a range of rates"""
    for i in range(start, stop):
        X[i-(3*RESOLUTION)] = i / RESOLUTION
        Y[i-(3*RESOLUTION)] = lyapunov(i / RESOLUTION)


if __name__ == '__main__':

    X = mp.Array('d', 1000)
    Y = mp.Array('d', 1000)

    CPU1 = mp.Process(target=loop, args=(3000, 3100, X, Y))
    CPU2 = mp.Process(target=loop, args=(3100, 3200, X, Y))
    CPU3 = mp.Process(target=loop, args=(3200, 3300, X, Y))
    CPU4 = mp.Process(target=loop, args=(3300, 3400, X, Y))
    CPU5 = mp.Process(target=loop, args=(3400, 3500, X, Y))
    CPU6 = mp.Process(target=loop, args=(3500, 3600, X, Y))
    CPU7 = mp.Process(target=loop, args=(3600, 3700, X, Y))
    CPU8 = mp.Process(target=loop, args=(3700, 3800, X, Y))
    CPU9 = mp.Process(target=loop, args=(3800, 3900, X, Y))
    CPU10 = mp.Process(target=loop, args=(3900, 4000, X, Y))

    CPU1.start()
    CPU2.start()
    CPU3.start()
    CPU4.start()
    CPU5.start()
    CPU6.start()
    CPU7.start()
    CPU8.start()
    CPU9.start()
    CPU10.start()

    CPU1.join()
    CPU2.join()
    CPU3.join()
    CPU4.join()
    CPU5.join()
    CPU6.join()
    CPU7.join()
    CPU8.join()
    CPU9.join()
    CPU10.join()

    ZERO = np.array(range(len(X))) * 0

    plt.plot(X, ZERO)
    plt.plot(X, Y)
    plt.show()
    print(time.time() - START_TIME)
```

A.3 Logic Map

```
# -*- coding: utf-8 -*-
"""
Created on Thu Jan 30 14:11:24 2020

@author: 967869@swansea.ac.uk
"""

#Imports ---------------------------------------------------------------------
import numpy as np
import matplotlib.pyplot as plt

#Define constants ------------------------------------------------------------

X=[]
Y=[]

#Define variables ------------------------------------------------------------

rate = 0
population = 0.2

#Law of the land -------------------------------------------------------------

def Law(r):
    global population
    population = r * population * ( 1 - population )
    #population = r - population**2
    return population

#Define functions ------------------------------------------------------------

def Cycle(r):
    global population
    for i in range(20000):
        Law(r)
    for i in range(20000,21000):
        X.append(rate)
        Y.append(Law(r))
    population= 0.2

def Scenario():
    global rate
    for i in range(1,2000):
        Cycle(rate)
        rate=i/1000

Scenario()

# Create the plot
plt.plot(X,Y,'.')

#Show the plot
plt.show()
```

A.4 Cobweb

```
# -*- coding: utf-8 -*-
"""
```

```python
Created on Thu Jan 30 14:11:24 2020

@author: 967869@swansea.ac.uk
"""

#Imports -------------------------------------------------------------
import numpy as np
import matplotlib.pyplot as plt

#Define constants -----------------------------------------------------

X=[]
Y=[0]

#Define variables -----------------------------------------------------

rate = 4
population = 0.2

#Law of the land ------------------------------------------------------

def Law():
    global population
    population = rate * population * ( 1 - population )
    return population

#Define functions -----------------------------------------------------

def Cobweb():

    for i in range(1500):
        Law()

    for i in range(320):
        X.append(population)
        X.append(population)
        Law()
        Y.append(population)
        Y.append(population)

#cobweb ---------------------------------------------------------------

m=np.array(range(1001))
n=rate*(m/1000)*(1-(m/1000))
o=m/1000

Cobweb()
X.append(population)

plt.plot(m/1000,n)
plt.plot(m/1000,o)
plt.plot(X,Y)
plt.show()
```

A.5 Fourier

```python
# -*- coding: utf-8 -*-
"""
Created on Thu Jan 30 14:11:24 2020

@author: 967869@swansea.ac.uk
"""

#Imports -------------------------------------------------------------
import numpy as np
import matplotlib.pyplot as plt
from scipy.fftpack import fft, ifft

#Define constants -----------------------------------------------------

History=[]


#Define variables -----------------------------------------------------

rate = 2

```

```python



population = 0.5

#Law of the land --------------------------------------------------------------

def Law():
    global population
    population = rate * population * ( 1 - population )
    return population

#Define functions -------------------------------------------------------------

def Series():
    for i in range(10000):
        Law()

    for i in range(1000):
        History.append(population)
        Law()

#cobweb -----------------------------------------------------------------------

array=np.array(range(1000))

Series()



complexfourier = fft(History)
fourier = complexfourier.real

gradient=np.gradient(np.gradient(fourier))
plt.plot(array[2::],fourier[2::],label="Fourier")



plt.plot(array[2::],History[2::],label="Series")

#plt.plot(array[2::],gradient[2::],label="Gradient")
plt.legend()
plt.show()

def countpeaks():
    n=1
    for i in range(2,1000):
        if abs(gradient[i]) >= 0.01:
            print(n)
            n+=1
```

A.6 Transitivity Bar Chart

```python
# -*- coding: utf-8 -*-
"""
Created on Wed Apr 15 14:24:32 2020

@author: 967869@swansea.ac.uk
"""

#Imports ----------------------------------------------------------------------
import numpy as np
import matplotlib.pyplot as plt

#Define constants -------------------------------------------------------------

X=[]
Y=[]

#Law of the land --------------------------------------------------------------

def Law(rate):
    population = 0.2
    list = []

    for i in range(2000000):
        population = rate * population * ( 1 - population )

    for i in range(50000):
        population = rate * population * ( 1 - population )
        rounded = round(population, 5)
        if rounded not in list:
            list.append(rounded)

    return len(list)

X.append(0.5)
Y.append(Law(0.5))

X.append(1)
Y.append(Law(1))

X.append(1.5)
Y.append(Law(1.5))

X.append(2)
Y.append(Law(2))

X.append(2.5)
Y.append(Law(2.5))

for i in range(30):
    X.append(2.98+(i/60))
    Y.append(Law(2.98+(i/60)))

for j in range(1000):
    X.append(3.49+(j/2000))
    Y.append(Law(3.49+(j/2000)))

plt.bar(X, Y, edgecolor = 'green', width = 0.0001)
plt.show()
```

A.7 Dense in X max value graph

```python
# -*- coding: utf-8 -*-
"""
Created on Fri Apr 24 17:07:10 2020

@author: 967869@swansea.ac.uk
"""


#Imports ------------------------------------------------------------------
import numpy as np
import matplotlib.pyplot as plt

#Define constants ------------------------------------------------------------

X=[]
Y=[]

#Define variables ------------------------------------------------------------

rate = 0
population = 0.2

#Law of the land -------------------------------------------------------------

def Law(r):
    global population
    population = r * population * ( 1 - population )
    #population = r - population**2
    return population

#Define functions ------------------------------------------------------------

def Cycle(r):
    global population
    for i in range(2000):
        Law(r)
    for i in range(2000,2100):
        X.append(population)
        Y.append(Law(r))
    population= 0.2
```

```python

def Scenario():
    global rate
    for i in range(1,4000):
        Cycle(rate)
        rate=i/1000

#Run this bad boy -----------------------------------------------------------

Scenario()

# Create the plot
plt.plot(X,Y,'.')

#Show the plot
plt.show()
```

A.8 Feigenbaums Constant

```python
# -*- coding: utf-8 -*-
"""
Created on Tue Apr 21 15:54:29 2020

@author: 967869@swansea.ac.uk (Inspired by Rosetta Code)
"""

max_it = 25          # For easier control over cycle length
max_it_k = 100        #
mu1 = 1           # This is Growth rate mu^(i-1)
mu2 = 0.0            # mu^(i-2)
f = 4.92            # Best approx. of Feigenbaums constant

print(" i        f          mu")
print("{0:2d}    {1:.8f}      {2:.5f}".format(1, f, mu1))

for i in range(2, max_it + 1):
    mu = mu1 + (mu1 - mu2) / f    # mu is mu^(i)
    for k in range(1, max_it_k + 1):
        p = 0.0
        dp = 0.0
        for n in range(1, (1 << i) + 1):
            dp = 1.0 - 2.0 * dp * p
            p = mu - p**2
        mu = mu - p / dp
    f = (mu1 - mu2) / (mu - mu1)

    print("{0:2d}    {1:.8f}      {2:.5f}".format(i, f, mu))
    mu2 = mu1
    mu1 = mu
```