FACULTY OF SCIENCE, TECHNOLOGY AND MEDICINE

University of Luxembourg

Faculty of Science, Technology and Medicine

Bachelor in Mathematics

Summer semester 2020/21

Lët'z box-counting!

Bachelor Thesis

Bachelor thesis submitted for the partial fulfilment of the degree in mathematics.

Written by
Kim Da Cruz

Supervised by
Guendalina Palmirotta
Lara Daw

3rd June 2021

Abstract

In this thesis, our goal is to find the box-dimension of the borders of Luxembourg. In order to do this, we start by introducing the Coastline paradox, that explains the reason why we work with fractal dimensions. Then, we will introduce one particular fractal dimension, namely the Hausdorff dimension. After that, we will define the box-counting dimension and prove some of its important properties. Finally, we will look at the box counting method, which will allow us to complete our goal.

Acknowledgements

First of all, I would like to thank my two supervisors, Guendalina Palmirotta and Lara Daw for suggesting this interesting topic. Thank you for answering all my questions that I had while working on my thesis and for helping me explore this world of fractals that is new to me.
I would also like to thank my family and friends, especially my parents and my brothers who supported me and helped me get a clean head when I needed a break.

Contents

1 Introduction

Almost everyone has already travelled in their life, if it is within their country, their town, or around the whole world. Clearly, every country has its own borders and a well known length, which goes from south to north, and width, from west to east, as well as an approximation of the area. However, what is not so well known is the length of the borders. The goal of this thesis will be to answer the following questions: *How can this length be determined and what is the length of the borders of Luxembourg?*
In order to answer these questions, we need to work with fractal dimension. One may ask why we cannot measure it using usual methods, for example by taking a ruler or anything similar and woking in the dimensions like meters or kilometres. The Coastline paradox, that will be introduced in the third section, explains why we cannot use the ordinary dimensions (like metres or miles, etc.) and why we need to work with fractals.

In the first paragraph, we stated the goal of this thesis, which is to find the length of the borders of Luxembourg. However, the Coastline paradox explains that this is impossible. Therefore, we set our goal to finding the fractal dimension of the borders of Luxembourg, more specifically, the box-counting dimension. In fact, one can notice that the name of this thesis was chosen from this goal. The part "Lët'z" comes from "Lëtzebuerg", but also from "Let's", so that the title can mean "Let's do box-counting" and "Luxembourg box-counting".

In Section 3, we will start by introducing the Coastline paradox and then move to the Hausdorff dimension. We will also mention the definition and some of its useful properties.

After being familiarised with fractal dimension, we come to Section 4, where we will define the box-counting dimension. In order to get comfortable with this dimension, we will explore some properties and prove them. We will also look at some examples and calculate the box dimension of some well-known curves, such as the Sierpiński triangle or the Koch curve. In the same section, we will also compare the two fractal dimensions that we mentioned before and notice that they have a lot of similar properties.

Lastly, in Section 5, we will look at a method, the box-counting method, that allows us to easily calculate the box dimension of given curves. We will implement two algorithms, one in Section 5.1 and the other one in Section 5.2, based on this method and afterwards, we will check if they are accurate, by applying them to some curves of which we know the dimension, for example a circle. Finally, we will use these algorithms to find the box dimension of the borders of Luxembourg.

2 Preliminaries

The aim of this section is to collect several definitions, explanations and results to rise the comprehension for the next sections. The following definitions can be found similarly in the book *Fractal Geometry: Mathematical Foundations and Applications* [1].

First, note that throughout this section, $\delta > 0$ is real and $n \in \mathbb{N}$. Next, we start with the definition of the diameter of a set, and follow with the definitions of a δ-cover and a δ-neighbourhood. These definitions will be useful to define and understand the box-counting dimension.

Definition 2.1. Let U be a non-empty subset of the n-dimensional Euclidean space $\mathbb{R}^n$. The *diameter* of U is defined as $|U| := \sup\{|x - y| : x, y \in U\}$, where $|x - y|$ denotes the Euclidean norm.

From this definition, we can now define a δ-cover and δ-neighbourhood of a given set.

Definition 2.2. Let F be a non-empty bounded subset of $\mathbb{R}^n$ and $\{U_i\}_{i\geq 1}$ be a countable or finite collection of sets. We say that $\{U_i\}_{i\geq 1}$ is a *δ-cover* of F if

(i) all the sets U_i, for $i \geq 1$, are of diameter at most δ, that is $0 < |U_i| \leq \delta$, and

(ii) they cover F, i.e. if $F \subset \bigcup_{i=1}^{\infty} U_i$.

Definition 2.3. Let F be a non-empty bounded subset of $\mathbb{R}^n$, the *δ-neighbourhood* F_δ of F is defined by

$$F_\delta = \{x \in \mathbb{R}^n : |x - y| \leq \delta \text{ for some } y \in F\},$$

that is, the set of points within distance δ of F.

In order to understand some of the properties of the box dimension, which we will look at later, let us also recall the Lebesgue measure on $\mathbb{R}^n$, as well as Hölder, Lipschitz and bi-Lipschitz transformations.

Definition 2.4. Let F be a subset of $\mathbb{R}^n$, then the *n-dimensional Lebesgue measure* $\mathcal{L}^n$ of F may be thought of as the extension of n-dimensional volume to a large class of sets. It is defined by

$$\mathcal{L}^n(F) = \inf \left\{ \sum_{i_1}^{\infty} vol^n(F_i) : F \subset \bigcup_{i=1}^{\infty} F_i \right\},$$

where vol^n denotes the n-dimensional volume, $\{F_i\}_{i\geq 1}$ is a covering of F and the infimum is taken over all coverings of F.

For any set F, for which the volume can be determined by usual rules of mensuration, we get that $\mathcal{L}^n(F) = vol^n(F)$.

Let us now recall what Hölder, Lipschitz and bi-Lipschitz transformations are. In order to do this, we follow closely the text in [6].

Definition 2.5. Let $A \subset \mathbb{R}^n$ and $B \subset \mathbb{R}^m$, for $m \in \mathbb{N}$. Let $|\cdot|$ denote the Euclidean distance in $\mathbb{R}^n$ respectively in $\mathbb{R}^m$. Consider the function $f : A \to B$.

(i) The function f is called *Hölder* continuous if there exists a real positive constant c and $\alpha > 0$ such that, for all x, $y \in A$

$$|f(x) - f(y)| \leq c|x - y|^\alpha.$$

(ii) In particular, if $\alpha = 1$, f is called *Lipschitz* continuous.

(iii) We say that f is *bi-Lipschitz* continuous, if there exists $c \geq 1$ such that

$$\frac{1}{c}|x - y| \leq |f(x) - f(y)| \leq c|x - y|.$$

Proposition 2.6. Let $A \subset \mathbb{R}^n$, $B \subset \mathbb{R}^m$ and $f : A \to B$ be a bi-Lipschitz transformation as in Definition 2.5. Then f is injective.

Proof. Let x, $y \in A$ such that $f(x) = f(y)$, then $f(x) - f(y) = 0$. This implies that $|f(x) - f(y)| = 0$. Since f is bi-Lipschitz, there exists $c \geq 1$ such that

$$\frac{1}{c}|x - y| \leq 0 \leq c|x - y|.$$

Hence $|x - y| = 0$, from which we conclude that $x = y$. $\square$

Lastly, let us recall some definitions that we will use in some properties later on.

Definition 2.7. ([7], [8], [9]). Let F be a non-empty subset of $\mathbb{R}^n$ and $f : \mathbb{R}^n \to \mathbb{R}^n$ be a function.

(a) We say that f is a similarity if it is a bijection that multiplies all distances by the same positive real number r, called the scaling factor, so that for any $a, b \in \mathbb{R}^n$, $|f(a) - f(b)| = r|a - b|$.

(b) We say that f is a congruence if it is distance preserving, i.e. if for any $a, b \in \mathbb{R}^n$, $|f(a) - f(b)| = |a - b|$.

(c) We say that f is an affine transformation if it preserves lines and parallelism but not necessarily distances and angles.

3 Introduction to fractal dimension

As explained in the introduction, we cannot measure the coastline or the borders of a country with our usual tools, such as rulers. The Coastline paradox, here below, explains why we need to consider fractal dimensions.

3.1 The Coastline Paradox

The inspiration for this section was found in [2], [3] and [4]. The Coastline paradox is the observation that the length of a coastline is not well-defined, it is impossible to determine an exact length. In fact, if we try to measure the coastline of a country, we get a different result for every method we use. For instance, if we use a ruler on a globe, we get a different length than if we measure it with a smaller ruler on the same globe.

It is easy to observe that the smaller the ruler, the longer the length of the coastline. But not only the ruler has an influence on the length we finally get, it also depends on the distance to the coastline. The closer we are, the more curves we see on the borders, and hence, the longer the coastline is.

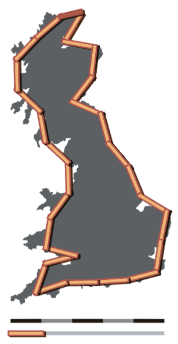


Figure 1: Measuring the coastline of Britain with a *bigger* ruler, [2].

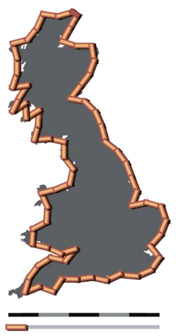


Figure 2: Measuring the coastline of Britain with a *smaller* ruler, [2].

In the figures above, we first measure the coastline of Great Britain using a 100km long ruler (Figure 1) and get a length of 2800km, then we measure it using a smaller ruler (Figure 2), of length 50km and obtain a coast length of 3400km (see [2]).

The reason why there is no obvious size of a coastline is that a landmass has different features at all scales. From far away, some coastlines seem to be lines but when approaching them, we see that there are more and more details. A landmass has features from thousands of kilometres in size, but also ones from fractions of millimetres or less; it all depends on the point of view.

One could think that the length we measure converges to a certain number if we get closer, but that is not the case. It is not like, for a metal bar for example, where we can get an upper and a lower bound for the length of it. In fact, the closer we get or the smaller ruler we use, the longer the length will be, so that there is no maximum length of a coastline.

This phenomenon was first observed by Lewis Fry Richardson, when trying to check that the likelihood of a war depends on the length of the borders between two countries. He also observed that when measuring more curvy coastlines, the rate at which the length grew was faster than the rate at which the length of smoother coastlines grew. This means that, for example the rate at which the length of the coastline of Great Britain grows is much faster than the rate at which the length of South Africa grows. This rate later became known as the fractal dimension.
Lewis Fry Richardson, not only, discovered that geographic borders are a fractal curve. After having collected data from several examples, he found out that the length, denoted by $L(S)$, of a geometric border can be approximated by a function of the form

$$L(S) \simeq CS^{1-D}, \tag{1}$$

where S is the measurement scale, C is a positive constant and $D \geq 1$ is a constant, called the dimension. From the observations above, smoother coastlines have a smaller fractal dimension, so that for these types of countries, the fractal dimension is close to 1. In Richardson's research, he found, by applying the above formula, that South Africa, which is a country with a smooth coastline, has a fractal dimension of 1.02, whereas Britain, who has curvier borders, has a dimension of 1.25.

Now, before calculating the fractal dimension of Luxembourg, we need to look at some definitions. For starters, there are a lot of types of fractal dimensions, such as the Hausdorff dimension and the box-counting dimension. Let us first introduce the Hausdorff dimension and afterwards we will analyse the box-counting dimension, in order to finally calculate the fractal dimension using the box-counting method.

3.2 Hausdorff measure and dimension

In this section we will look at one of the fractal dimensions, the *Hausdorff dimension*. The proofs of the following properties will be left out, however, if the reader is interested, they can be found in the book *Fractal Geometry - Mathematical Foundations and Applications (Third edition)*, [1] in Part I, Section 3.

3.2.1 Hausdorff measure

In order to define the Hausdorff dimension, let us first define the Hausdorff measure.

Definition 3.1. Let F be a non empty subset of $\mathbb{R}^n$ and $s \geq 0$, then for any $\delta > 0$, we define

$$\mathcal{H}^s_\delta(F) := \inf \left\{ \sum_{i=1}^{\infty} |U_i|^s : \{U_i\}_{i \geq 1} \text{ is a } \delta\text{-cover of } F \right\}. \tag{2}$$

In this definition, the infimum is taken over the sum of the s^{th} power of the sum of the diameters. We see that, if $0 < \epsilon < \delta$, then the δ-covers of F consist of every cover of F of size at most δ, thus also those of size ϵ or smaller. And for ϵ-covers, we only consider the covers of size smaller than ϵ. Hence, as δ decreases, the number of δ-covers of F also decreases and so does the sum of the s^{th} power of the diameters. This implies that if $\delta \to 0$, then the infimum decreases, or at least it doesn't increase. Thus, the infimum approaches a limit as δ goes to 0. This observation leads us to the definition of the s-dimensional Hausdorff measure of F, given by

$$\mathcal{H}^s(F) = \lim_{\delta \to 0} \mathcal{H}^s_\delta.$$

This limit exists for any subset F of $\mathbb{R}^n$.

Proposition 3.2 (Scaling property)**.** Let $F \subset \mathbb{R}^n$ and $f : \mathbb{R}^n \to \mathbb{R}^n$ be a similarity transformation of scale factor $\lambda > 0$, then for $s \geq 0$

$$\mathcal{H}^s(f(F)) = \lambda^s \mathcal{H}^s(F).$$

The complete proof of this proposition can be found in ([1], Scaling property 3.2, p.46).

3.2.2 Hausdorff dimension

From the definition of the Hausdorff measure, we know that for any non-empty subset F of $\mathbb{R}^n$ and for any $0 < \delta < 1$, $\mathcal{H}^s_\delta$ is non-increasing which implies that the Hausdorff measure $\mathcal{H}^s$ is also non-increasing. In fact we can even see that, for any δ-cover $\{U_i\}_{i \geq 1}$ of F, $s \geq 0$ and any $t > s$,

$$\sum_{i=0}^{\infty} |U_i|^t = \sum_{i=0}^{\infty} |U_i^{t-s} U_i^s| \leq \sum_{i=0}^{\infty} |U_i|^{t-s} |U_i|^s \leq \delta^{t-s} \sum_{i=0}^{\infty} |U_i|^s.$$

The second inequality follows from the fact that the diameter of the δ-covers of F is at most δ. Applying the infimum and taking the limit as δ goes to 0, we obtain

$$\lim_{\delta \to 0} \mathcal{H}^t_\delta(F) \leq \mathcal{H}^s(F) \lim_{\delta \to 0} \delta^{t-s}.$$

This again implies that if the s-dimensional Hausdorff measure of F, $\mathcal{H}^s(F)$ is finite, then $\mathcal{H}^t = 0$.

Hence, we notice that there is a critical value of s, for which $\mathcal{H}^s$ falls from '∞' to '0', see Figure 3. This critical value is called the *Hausdorff dimension* and denoted $\dim_H F$.

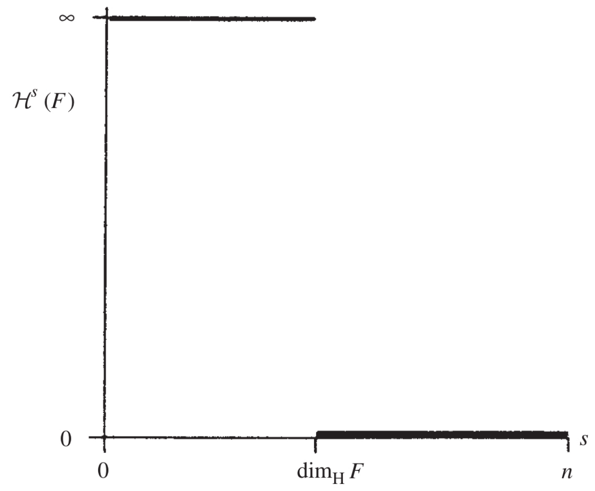


Figure 3: Graph of $\mathcal{H}^s$ against s, ([1] p. 48).

Definition 3.3. Let F be a subset of $\mathbb{R}^n$, $s \geq 0$ and $\mathcal{H}^s(F)$ be the Hausdorff measure of F. Then the Hausdorff dimension of F is defined by

$$\dim_H F = \inf \{s \geq 0 : \mathcal{H}^s(F) = 0\} = \sup \{s \geq 0 : \mathcal{H}^s(F) = \infty\}.$$

We also see that

$$\mathcal{H}^s(F) = \begin{cases} \infty & \text{if } 0 \leq s < \dim_H F, \\ 0 & \text{if } s > \dim_H F. \end{cases}$$

Furthermore, if $s = \dim_H F$, then $\mathcal{H}^s(F)$ can be '0', '∞' or $0 < \mathcal{H}^s(F) < \infty$.

Now, let us mention some important properties of the Hausdorff dimension. The complete proofs of the following propositions can found in ([1], starting on page 48.)

Proposition 3.4. Let $n \in \mathbb{N}$ and E and F be two non-empty subsets of $\mathbb{R}^n$, then the Hausdorff dimension has the following properties.

(a) *Monotonicity*: If $E \subset F$, then $\dim_H E \leq \dim_H F$.

(b) *Range of values*: $0 \leq \dim_H F \leq n$.

(c) *Countable stability*: Let $\{F_n\}_{n\geq 1}$ be a countable sequence of non-empty subsets of $\mathbb{R}^n$, then

$$\dim_H \bigcup_{i=1}^{\infty} F_i = \sup_{1\leq i<\infty} \{\dim_H F_i\}.$$

(d) *Countable sets*: If F is countable, then $\dim_H F = 0$.

(e) *Open sets*: If $F \subset \mathbb{R}^n$ is open, then $\dim_H F = n$.

Proposition 3.5. Let $m \in \mathbb{N}$.

(i) If $F \subset \mathbb{R}^n$ is non-empty and $f : F \to \mathbb{R}^m$ is Hölder continuous, with Hölder exponent $\alpha > 0$, then

$$\dim_H f(F) \leq \frac{1}{\alpha} \dim_H F.$$

In particular, if f is a Lipschitz transformation, i.e. if $\alpha = 1$, then $\dim_H f(F) \leq \dim_H F$.

(ii) If $F \subset \mathbb{R}^n$ and $f : F \to \mathbb{R}^m$ is a bi-Lipschitz transformation, then

$$\dim_H f(F) = \dim_H F.$$

From this proposition follow the next properties.

Proposition 3.6. Let F be a non-empty subset of $\mathbb{R}^n$ and $m \in \mathbb{N}$. In order to understand one of the properties, we recall that a smooth manifold is an infinitely differentiable curve or surface.

(a) *Geometric invariance*: If $f : F \to \mathbb{R}^m$ is a congruence, similarity or affine transformation, then $\dim_H f(F) = \dim_H F$.

(b) *Smooth curves*: If F is a smooth m-dimensional manifold (e.g. curve, surface,...), then $\dim_H F = m$.

(c) *Projections*: If $F \subset \mathbb{R}^2$, then $\dim_H \mathit{proj}(F) \leq \min\{1, \dim_H F\}$, where *proj* denotes the orthogonal projection from $\mathbb{R}^2$ onto some given line through the origin.

Example 3.7. Let F be the middle third Cantor set, then $\dim_H F = \frac{\log(2)}{\log(3)}$, with $\frac{1}{2} \leq \mathcal{H}^s(F) \leq 1$ if $s = \frac{\log(2)}{\log(3)}$.

Before proving the result, let us recall the construction of the middle third Cantor set. In fact, it is constructed by deleting the middle third part of intervals, starting with the unit interval. More precisely, let $E_0 = [0, 1]$ be the unit interval, then E_1 is obtained by deleting the middle third part of this interval, so that $E_1 = [0,\, \frac{1}{3}] \cup [\frac{2}{3},\, 1]$. To obtain E_2, we delete the middle third part of the two intervals of E_1, thus E_2 consists of the four intervals $[0,\, \frac{1}{9}]$, $[\frac{2}{9},\, \frac{1}{3}]$, $[\frac{2}{3},\, \frac{7}{9}]$ and $[\frac{8}{9},\, 1]$. By iteration, for any integer k, E_k consists of 2^k intervals of length 3^{-k}.

The *middle third Cantor set* F consists of all the numbers that are in every interval of every E_k, precisely, $F = \bigcap_{k=0}^{\infty} E_k$. In fact, F can be seen as the limit of the E_k's as k tends to infinity. However, it is impossible to draw F because of its infinitesimal detail. This explains why we illustrate F, by drawing pictures of one of the E_k, where k is any integer, see Figure 4 below.

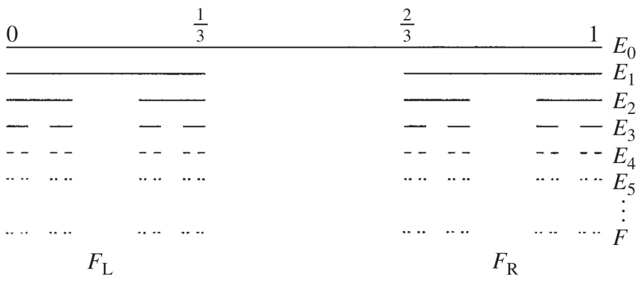


Figure 4: Construction of the middle third Cantor set F, ([1] p. xx).

- Let us now show that $\dim_H F = \frac{\log(2)}{\log(3)}$.
  In order to do this, we split F into a left part $F_L := F \cap [0,\, \frac{1}{3}]$ and a right part $F_R := F \cap [\frac{2}{3},\, 1]$. Clearly, $F = F_L \dot{\cup} F_R$, where $\dot{\cup}$ denotes the disjoint union. This implies that

$$\mathcal{H}^s(F) = \mathcal{H}^s(F_L) + \mathcal{H}^s(F_R), \quad \text{for any } s \geq 0.$$

  In addition, by construction of F, F_L and F_R are similar to F but scaled at ratio $\lambda = \frac{1}{3}$. From the scaling property, Proposition 3.2, it follows that

$$\mathcal{H}^s(F_L) + \mathcal{H}^s(F_R) = \left(\frac{1}{3}\right)^s \mathcal{H}^s(F) + \left(\frac{1}{3}\right)^s \mathcal{H}^s(F).$$

  Hence, we conclude that

$$\mathcal{H}^s(F) = 2\left(\frac{1}{3}\right)^s \mathcal{H}^s(F).$$

  For $s = \dim_H F$ we want to find a value of s such that $0 < \mathcal{H}^s(F) < \infty$, so we can divide by $\mathcal{H}^s(F)$ and obtain

$$1 = 2\left(\frac{1}{3}\right)^s \iff \frac{1}{2} = \left(\frac{1}{3}\right)^s \iff \log\left(\frac{1}{2}\right) = s\log\left(\frac{1}{3}\right) \iff s = \frac{\log(2)}{\log(3)}.$$

Hence, $\dim_H F = \frac{\log(2)}{\log(3)}$.

- Next, we will show that $\mathcal{H}^s(F) \leq 1$.
  Note that at each level of construction $k \in \mathbb{N}$, the set E_k consists of 2^k intervals of length 3^{-k}, denoted by $I_1, I_2, ..., I_{2^k}$. Let $\delta > 0$, then if we chose k such that $\delta = 3^{-k}$, then the intervals I_1, $...I_{2^k}$ of E_k are δ-covers of F. This gives

$$\mathcal{H}^s_\delta(F) = \inf_{\delta\text{-covers of } F} \left\{ \sum_{i=1}^{\infty} |U_i|^s : \{U_i\}_{i\geq 1} \text{ is a } \delta\text{-cover of } F \right\} \leq \sum_{i=1}^{2^k} |I_i|^s = 2^k 3^{-ks}.$$

  For $s = \frac{\log(2)}{\log(3)}$, we have $2^k 3^{-k\frac{\log(2)}{\log(3)}} = 2^k e^{-k\frac{\log(2)}{\log(3)}\log(3)} = e^{k\log(2)} e^{-k\log(2)} = 1$. Hence, $\mathcal{H}^s_\delta(F) \leq 1$ and we conclude that

$$\mathcal{H}^s(F) = \lim_{\delta \to 0} \mathcal{H}^s_\delta(F) \leq 1.$$

- Finally, let us show that $\mathcal{H}^s(F) \geq \frac{1}{2}$.
  To this end, we will show that for any cover $\{U_i\}_{i\geq 1}$ of F, we have

$$\sum_{i=1}^{\infty} |U_i|^s \geq \frac{1}{2}.$$

  In fact, this implies that $\mathcal{H}^s_\delta(F) \geq \frac{1}{2}$, for any $\delta > 0$, and hence, $\lim_{\delta\to 0} \mathcal{H}^s_\delta(F) \geq \frac{1}{2}$.
  First, note that $\frac{1}{2} = 3^{-s}$ if $s = \frac{\log(2)}{\log(3)}$, therefore, showing that $\sum_{i=1}^{\infty} |U_i|^s \geq \frac{1}{2}$ is equivalent to prove that $\sum_{i=1}^{\infty} |U_i|^s \geq 3^{-s}$.

  Since F consists of intervals, we can assume that $\{U_i\}_{i\geq 1}$ is a collection of intervals. Let k be an integer satisfying $3^{-k-1} \leq |U_i| < 3^{-k}$ for any $i \geq 1$ and consider the intervals of E_k. We see that each U_i can intersect at most one of the intervals of E_k since the distance between these intervals is at least 3^{-k}.

  Let $j \geq k$ and consider the intervals of E_j. By construction, each U_i intersects at most $\frac{2^j}{2^k}$ intervals of E_j. That is, at most $2^{j-k} = 2^j 3^{-sk} = 2^j 3^{s+s(-k-1)} \leq 2^j 3^s |U_i|^s$ intervals. We also notice that all the U_i's together intersect all the 2^j intervals of E_j. So that summing the number of intervals that they intersect together is 2^j. Hence, we obtain

$$2^j \leq \sum_{i=1}^{\infty} 2^j 3^s |U_i|^s = 2^j 3^s \sum_{i=1}^{\infty} |U_i|^s.$$

  We conclude by dividing by $2^j 3^s > 0$ that $3^{-s} \leq \sum_{i=1}^{\infty} |U_i|^s$.

4 Box-counting dimension

The following notes are again inspired and based from [1], Sections 2.1 and 2.2. Throughout this part of the document, we set $n \in \mathbb{N}$ and $\delta \in \mathbb{R}_+^*$.

4.1 Definitions and Properties

In this section, we will define the box-counting dimension and look at its properties and later, use it to calculate the dimension of the borders of Luxembourg.

Given a non-empty subset F of $\mathbb{R}^n$ and, for each $\delta > 0$, let $N_\delta(F)$ denote the δ-covers of F. If

$$N_\delta(F) \simeq c\delta^{-s}, \tag{3}$$

where c is a constant and $s \geq 0$, we observe that (3) is very similar to Lewis Fry Richardson's observation (1) on the approximation for fractal dimension. Moreover, s represents the so-called box-dimension of F.

Definition 4.1. Let F be a non-empty bounded subset of $\mathbb{R}^n$, and let $N_\delta(F)$ be the smallest number of sets of diameter at most δ which can cover F, i.e. the least number of sets in any δ-cover of F. We define

(a) the *lower box-counting dimension* of F as

$$\underline{\dim}_{\mathrm{B}} F = \varliminf_{\delta \to 0} \frac{\log(N_\delta(F))}{-\log(\delta)}, \tag{4}$$

(b) the *upper box-counting dimension* of F as

$$\overline{\dim}_{\mathrm{B}} F = \varlimsup_{\delta \to 0} \frac{\log(N_\delta(F))}{-\log(\delta)}. \tag{5}$$

Clearly, since 'lim inf' of any function is always smaller than 'lim sup' of the same function, $\underline{\dim}_{\mathrm{B}} F \leq \overline{\dim}_{\mathrm{B}} F$. If $\underline{\dim}_{\mathrm{B}} F = \overline{\dim}_{\mathrm{B}} F$, we define the *box-counting dimension* or *box dimension* of F by

$$\dim_B F = \lim_{\delta \to 0} \frac{\log(N_\delta(F))}{-\log(\delta)}. \tag{6}$$

Remark 4.2. By the following, let us assume that δ is sufficiently small to ensure that $-\log(\delta) > 0$. In addition, we only consider box-dimension of non-empty bounded sets, otherwise $\log(N_\delta(F))$ could be equal to '$\log(0)$' or '$\log(\infty)$'.

Definition 4.3. The family of cubes of the form

$$[m_1\delta,\ (m_1+1)\delta] \times \ldots \times [m_n\delta,\ (m_n+1)\delta],$$

where m_1, m_2, ..., m_n are integers. is called the *δ-mesh* or *δ-grid* of $\mathbb{R}^n$.
In other words, a δ-mesh in $\mathbb{R}^n$ is a family of cubes of side δ, that don't intersect each other. Since these cubes can form a grid, they are also called a δ-grid of $\mathbb{R}^n$.
Let us illustrate this definition by an example in $\mathbb{R}^2$.

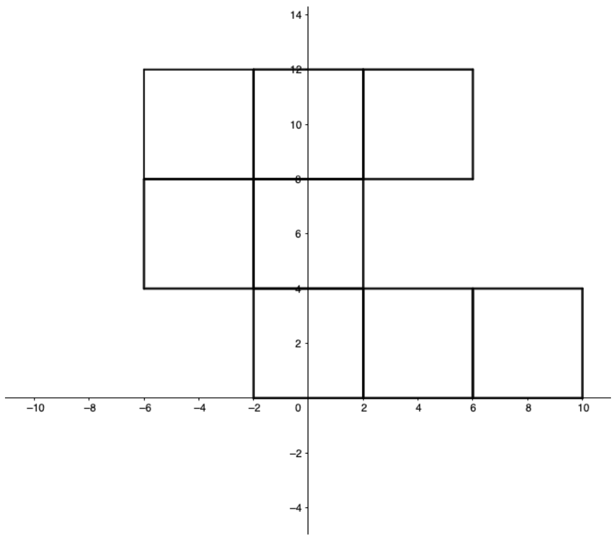


Figure 5: δ-mesh cubes in $\mathbb{R}^2$ with $\delta = 2$.

Proposition 4.4. In Definition 4.1, $N_\delta(F)$ can be any of the following:

(i) the smallest number of sets of diameter at most δ that cover F;

(ii) the smallest number of closed balls of radius δ that cover F;

(iii) the smallest number of cubes of side δ that cover F;

(iv) the number of δ-mesh cubes that intersect F;

(v) the largest number of disjoint balls of radius δ with centres in F.

Proof. Since the proofs of the equivalences are all similar, we will only focus on the following.
(i) $\iff$ (iv): Let $N_\delta(F)$ be the smallest number of sets of diameter at most δ that cover F and $N'_\delta(F)$ the number of δ-mesh cubes that intersect F.

We see that each δ-mesh cube that intersects F can be seen as a set of diameter $\delta\sqrt{n}$ that covers F. In particular, in $\mathbb{R}^2$, a square of side δ can be seen as a set with the same diameter as the square. By the Pythagoras theorem, this diameter is equal to $\delta\sqrt{2}$. Thus, we can construct at least $N'_\delta(F)$ sets of diameter $\delta\sqrt{n}$ that cover F, so that

$$N_{\delta\sqrt{n}}(F) \leq N'_\delta(F).$$

In addition, consider a set U of diameter δ that covers F. Any point in this set is contained in a δ-mesh cube C since these cubes intersect F. Hence, U is contained in the δ-mesh consisting of the cube C and its neighbouring cubes. Since the set U and the cube C were chosen randomly, this implies that any set of diameter at most δ that covers F is contained in 3^n δ-mesh cubes.

Thus, the number of δ-mesh cubes that intersect F is at most equal to the 3^n times the smallest number of sets of diameter δ that cover F, so that

$$N'_\delta(F) \leq 3^n N_\delta(F).$$

Combining these inequalities and applying the logarithm on every side as well as dividing by $-\log(\delta)$, we obtain

$$\frac{\log\left(N_{\delta\sqrt{n}}(F)\right)}{-\log(\delta)} \leq \frac{\log\left(N'_\delta(F)\right)}{-\log(\delta)} \leq \frac{\log\left(3^n N_\delta(F)\right)}{-\log(\delta)}.$$

We know that

$$-\log(\delta) = -\log\left(\frac{\delta\sqrt{n}}{\sqrt{n}}\right) = -\log(\delta\sqrt{n}) + \log(\sqrt{n}),$$

thus

$$\frac{\log\left(N_{\delta\sqrt{n}}(F)\right)}{-\log(\delta\sqrt{n}) + \log(\sqrt{n})} \leq \frac{\log\left(N'_\delta(F)\right)}{-\log(\delta)} \leq \frac{\log\left(3^n\right)\log\left(N_\delta(F)\right)}{-\log(\delta)}.$$

If $\delta \to 0$, one gets

- $-\log(\delta\sqrt{n}) + \log(\sqrt{n}) \to -\log(\delta\sqrt{n})$,
- $\delta\sqrt{n} \to \delta$ and
- $\frac{\log(3^n)}{-\log(\delta)} \to 0$ since $-\log(\delta) \to \infty$.

Hence, if we take the lower limits as $\delta \to 0$, we have

$$\varliminf_{\delta\to 0} \frac{\log\left(N_\delta(F)\right)}{-\log(\delta)} \leq \varliminf_{\delta\to 0} \frac{\log\left(N'_\delta(F)\right)}{-\log(\delta)} \leq \varliminf_{\delta\to 0} \frac{\log\left(N_\delta(F)\right)}{-\log(\delta)}.$$

We conclude that the definition of the lower box limit is verified for $N_\delta(F)$ and $N'_\delta(F)$, because $\underline{\lim}_{\delta\to 0} \frac{\log\left(N'_\delta(F)\right)}{-\log(\delta)} = \underline{\lim}_{\delta\to 0} \frac{\log(N_\delta(F))}{-\log(\delta)}$. Similarly, if we take the upper limit instead of the lower limit in the last step, we conclude that in Definition 4.1 (b), we can substitute $N_\delta(F)$ by $N'_\delta(F)$.

(i) $\iff$ (v): Under the same notations as above, $N_\delta(F)$ is the smallest number of δ-covers, and let $N''_\delta(F)$ be the largest number of disjoint balls of radius δ with centres in F. Moreover, let $\{B_1, B_2, ..., B_{N''_\delta(F)}\}$ be a collection of disjoint balls of radius δ with centres in F.

Let $x \in F$, then the distance between x and any of the B_i's, for $i \in \{1, 2, ..., N''_\delta(F)\}$ is at most δ. Otherwise, we could construct a ball of radius δ with centre x that is disjoint to the other balls. This again would imply, that there is a collection of more than $N''_\delta(F)$ balls with centres in F.

Hence, the $N''_\delta(F)$ non-disjoint balls of diameter 4δ that are concentric with the B_i's, cover F. Thus, the number of sets of diameter 4δ that cover F, is at most equal to the number of disjoint balls of radius δ with centres in F, so that

$$N_{4\delta} \leq N''_\delta(F).$$

In addition, let $\{U_1, U_2, ..., U_k\}$ be a collection of sets of diameter at most δ that cover F, then by definition of $N_\delta(F)$, $k \geq N_\delta(F)$.
Since the sets cover F and the balls have centres in F, for $i \in \{1, 2, ..., N''_\delta(F)\}$ and $j \in \{1, 2, ..., k\}$, each ball B_i contains at least one set U_j. Moreover, the B_i's are all disjoint, which implies that there are at least as many sets as balls, so that

$$N''_\delta(F) \leq N_\delta(F).$$

Just as in the proof of the previous equivalence, we combine the inequalities, apply the logarithm and divide by $-\log(\delta)$ to obtain

$$\frac{\log\left(N_{4\delta}(F)\right)}{-\log(\delta)} \leq \frac{\log\left(N''_\delta(F)\right)}{-\log(\delta)} \leq \frac{\log\left(N_\delta(F)\right)}{-\log(\delta)}.$$

Doing similar calculations as before and taking the lower limits as $\delta \to 0$, we get

$$\underline{\lim}_{\delta\to 0} \frac{\log\left(N_\delta(F)\right)}{-\log(\delta)} \leq \underline{\lim}_{\delta\to 0} \frac{\log\left(N''_\delta(F)\right)}{-\log(\delta)} \leq \underline{\lim}_{\delta\to 0} \frac{\log\left(N_\delta(F)\right)}{-\log(\delta)}.$$

We conclude as before that in Definition 4.1 (a) and (b) it we can either choose $N_\delta(F)$ or $N''_\delta(F)$. □

Now, let us illustrate Definition 4.1 by some examples.

Example 4.5. Let F be the *middle third Cantor set* and let us calculate $\underline{\dim}_B F$ and $\overline{\dim}_B F$.
For a recall on the construction of this set, see Example 3.7. Now, in order to do calculate the box dimensions of this set, let us choose δ such that $3^{-k} < \delta \leq 3^{-k+1}$, $\forall k \in \mathbb{N}$. We see that at each level of construction of F, there are 2^k intervals of length 3^{-k}. We also notice that all those intervals cover F. Thus, these intervals provide a δ-cover of F. Let $N_\delta(F)$ be the smallest number of sets of diameter at most δ that cover F, then it is easy to see that $N_\delta(F) \leq 2^k$.

In addition, if we take the (upper or lower) limit, as $\delta \to 0$, then $k \to \infty$ since $3^{-k} < \delta \leq 3^{-k+1}$. In conclusion, we obtain, by Definition 4.1 (b), that

$$\overline{\dim}_B F = \overline{\lim_{\delta\to 0}} \frac{\log(N_\delta(F))}{-\log(\delta)} \leq \overline{\lim_{k\to\infty}} \frac{\log(2^k)}{-\log(3^{-k+1})} = \overline{\lim_{k\to\infty}} \frac{k\log(2)}{(k-1)\log(3)} = \frac{\log(2)}{\log(3)}.$$

On the other hand, if we choose δ, so that $3^{-k-1} \leq \delta < 3^{-k}$, then every interval of length $\delta\,(< 3^{-k})$ intersects **at most** one of the intervals at the level of construction E_k. This is because the gap between the intervals is at leat 3^{-k}.

Since there are 2^k intervals that cover F at each level, we need **at least** 2^k intervals of length δ to cover F. This implies that $N_\delta(F) \geq 2^k$, so that by Definition 4.1 (a)

$$\underline{\dim}_B F = \underline{\lim_{\delta\to 0}} \frac{\log(N_\delta(F))}{-\log(\delta)} \geq \underline{\lim_{k\to\infty}} \frac{\log(2^k)}{-\log(3^{-k-1})} = \underline{\lim_{k\to\infty}} \frac{k\log(2)}{(k+1)\log(3)} = \frac{\log(2)}{\log(3)}.$$

Thus,

$$\overline{\dim}_B F \leq \frac{\log(2)}{\log(3)} \leq \underline{\dim}_B F.$$

However, by definition $\overline{\dim}_B F \geq \underline{\dim}_B F$, so we conclude that

$$\overline{\dim}_B F = \underline{\dim}_B F = \frac{\log(2)}{\log(3)}.$$

Example 4.6. Let F be the Sierpiński triangle with side length 1. Before calculating its box dimension, we will recall what the Sierpiński triangle is.

It is constructed similarly as the middle third Cantor set. The difference is that here we start with an equilateral triangle with unit length and remove the inverted equilateral triangle in the middle, see Figure 6 below. More precisely, let E_0 be the equilateral triangle of side length 1, then E_1 is obtained by removing the inverted equilateral triangle. This implies that E_1 consists of three equilateral triangles with side length $\frac{1}{2}$. To obtain E_2, we remove the inverted equilateral triangle in the three triangles of E_1, so that E_2 consists of nine equilateral triangles of side length $\frac{1}{4}$.

By iteration, for any integer k, E_k consists of 3^k triangles of side length $\frac{1}{2^k}$. The *Sierpiński triangle* is the intersection of all the triangles that are in every E_k.

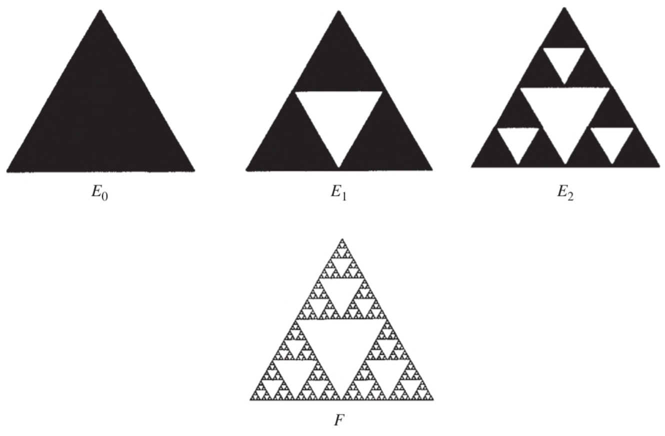


Figure 6: Construction of the Sierpiński triangle, ([1] p. xxii).

Now, let us calculate $\underline{\dim}_{\mathrm{B}} F$ and $\overline{\dim}_{\mathrm{B}} F$. First, we notice that since, for every E_k, there are 3^k triangles of side length 2^{-k} that cover F, 3^k sets of diameter at most δ provide a δ-cover of F, where δ is chosen such that $2^{-k} < \delta \leq 2^{-k+1}$. Hence, it is easy to see that $N_\delta(F) \leq 3^k$.

As before, it is easy to check that, if $\delta \to 0$, then $k \to \infty$, thus by Definition 4.1 (b), we obtain

$$\overline{\dim}_{\mathrm{B}} F = \overline{\lim_{\delta\to 0}} \frac{\log(N_\delta(F))}{-\log(\delta)} \leq \overline{\lim_{k\to\infty}} \frac{\log(3^k)}{-\log(2^{-k+1})} = \overline{\lim_{k\to\infty}} \frac{k\log(3)}{(k-1)\log 2} = \frac{\log(3)}{\log(2)}.$$

On the other hand, since only three triangles at a time have a distance that is smaller than 2^{-k}, if we chose δ satisfying $2^{-k-1} \leq \delta < 2^{-k}$, we see that a set of diameter δ intersects **at most** three triangles. Since, at each stage of construction there are 3^k triangles that cover F, we need **at least** $\frac{3^k}{3}$ sets of diameter at most δ to cover F. This implies, that $N_\delta(F) \geq 3^{k-1}$, so that by Definition 4.1 (a),

$$\underline{\dim}_{\mathrm{B}} F = \underline{\lim_{\delta\to 0}} \frac{\log(N_\delta(F))}{-\log(\delta)} \geq \underline{\lim_{k\to\infty}} \frac{\log(3^{k-1})}{-\log(2^{-k-1})} = \underline{\lim_{k\to\infty}} \frac{(k-1)\log(3)}{(k+1)\log 2} = \frac{\log(3)}{\log(2)}.$$

Thus,

$$\overline{\dim}_{\mathrm{B}} F \leq \frac{\log(3)}{\log(2)} \leq \underline{\dim}_{\mathrm{B}} F.$$

However, by definition $\overline{\dim}_{\mathrm{B}} F \geq \underline{\dim}_{\mathrm{B}} F$, so we conclude that

$$\overline{\dim}_{\mathrm{B}} F = \underline{\dim}_{\mathrm{B}} F = \frac{\log(3)}{\log(2)}.$$

Example 4.7. Let us calculate $\underline{\dim}_{\mathrm{B}} F$ and $\overline{\dim}_{\mathrm{B}} F$, where F is the Koch curve.

The Koch curve is constructed by replacing at each step of construction the middle third part of every interval by the other two sides of an equilateral triangle. More precisely, let E_k be the curve at the k^{th} stage of construction, with $k \in \mathbb{N}$, we start with the unit interval $E_0 = [0,\, 1]$ and obtain E_1 by replacing the middle third part of this interval by two sides of an equilateral triangle. After that, E_2 is obtained by replacing in each interval of E_1 the middle third part. We continue that way, so that at each level of construction $k \in \mathbb{N}$, E_k consists of 4^k intervals of length 3^{-k}. However, because of its infinitesimal detail, it is impossible to draw F. That is why, we illustrate it by representing it by one of the E_k, where k is a positive integer, see Figure 7 below.

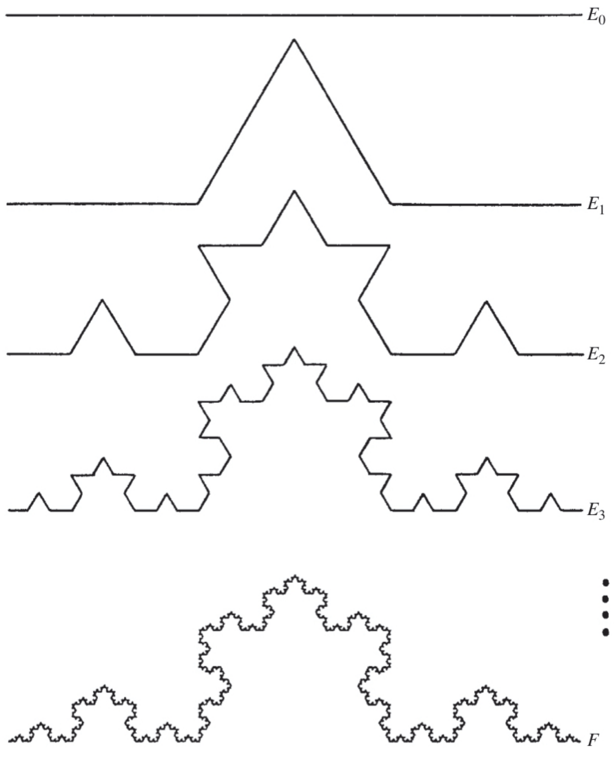


Figure 7: Construction of the Koch curve F, ([1] p. xxi).

Since, at each stage k of construction, E_k consists of 4^k intervals of length 3^{-k} that cover F, these intervals provide a δ-cover of F, if δ was chosen such that $3^{-k} < \delta \leq 3^{-k+1}$. Covering F with sets of diameter at most δ, makes it clear that

$N_\delta \leq 4^k$. By Definition 4.1 (b), we get

$$\overline{\dim}_{\mathrm{B}} F = \overline{\lim_{\delta\to 0}} \frac{\log(N_\delta(F))}{-\log(\delta)} \leq \overline{\lim_{k\to\infty}} \frac{\log(4^k)}{-\log(3^{-k+1})} = \overline{\lim_{k\to\infty}} \frac{k\log(4)}{(k-1)\log(3)} = \frac{\log(4)}{\log(3)}.$$

On the other hand, let us chose δ satisfying $3^{-k-1} \leq \delta < 3^{-k}$. Since the length of δ-mesh cubes is strictly less than 3^{-k}, each cube intersects at most one interval of length 3^{-k}. This implies that we need at least 4^k δ-mesh cubes to cover E_k. Since E_k covers F, this implies that $N_\delta(F) \geq 4^k$. By Definition 4.1 (a), this gives

$$\underline{\dim}_{\mathrm{B}} F = \underline{\lim}_{\delta\to 0} \frac{\log(N_\delta(F))}{-\log(\delta)} \geq \underline{\lim}_{k\to\infty} \frac{\log(4^k)}{-\log(3^{-k-1})} = \underline{\lim}_{k\to\infty} \frac{k\log(4)}{(k+1)\log(3)} = \frac{\log(4)}{\log(3)}.$$

We conclude similarly as in the previous examples and get

$$\overline{\dim}_{\mathrm{B}} F = \underline{\dim}_{\mathrm{B}} F = \dim_B F = \frac{\log(4)}{\log(3)}.$$

Remark 4.8. We see that in the first example, F is obtained by repeatedly copying 2 lines at scale $\frac{1}{3}$ and we found that $\dim_B F = \frac{\log(2)}{\log(3)}$. In the second example, F is obtained by repeatedly copying 3 triangles at scale $\frac{1}{2}$ and we found that $\dim_B F = \frac{\log(3)}{\log(2)}$. In the last example, we also repeatedly copied 4 lines at scale $\frac{1}{3}$ to construct F and obtained that $\dim_B F = \frac{\log(4)}{\log(3)}$. Thus we can assume that the box-counting dimension of a set F, that is obtained by repeatedly copying $m \in \mathbb{N}_{>1}$ similar disjoint figures at scale $r \in]0,1[$, is given by $\dim_B F = \frac{\log(m)}{\log(1/r)} = \frac{\log(m)}{-\log(r)}$.

After having applied the definition of the box-counting dimension to some examples, let us look at some of its properties and prove them.

Proposition 4.9. If F is a subset of $\mathbb{R}^n$, then

$$\underline{\dim}_{\mathrm{B}} F = n - \underline{\lim}_{\delta\to 0} \frac{\log(\mathcal{L}^n(F_\delta))}{\log(\delta)}$$
$$\overline{\dim}_{\mathrm{B}} F = n - \overline{\lim_{\delta\to 0}} \frac{\log(\mathcal{L}^n(F_\delta))}{\log(\delta)},$$

where F_δ is the δ-neighbourhood of F.

Proof. Consider a covering of F with balls of radius δ. Note that each point of the δ-neighbourhood F_δ of F is at distance at most δ from any point of F. Thus, the balls of radius 2δ, that are concentric with the $N_\delta(F)$ balls that cover F form a covering of F_δ. This implies that, if c_n is the volume of the unit ball in $\mathbb{R}^n$, the volume of F_δ is at most $c_n(2\delta)^n N_\delta(F)$.[1]

Hence, by Definition 2.4

$$\mathcal{L}^n(F_\delta) \leq c_n(2\delta)^n N_\delta(F).$$

Taking logarithms on each side and dividing by $-\log(\delta)$, we get

$$\begin{aligned}\frac{\log(\mathcal{L}^n(F))}{-\log\delta} \leq \frac{\log(N_\delta(F)c_n(2\delta)^n)}{-\log(\delta)} &= \frac{\log(2^n c_n) + n\log(\delta) + \log(N_\delta(F))}{-\log(\delta)} \\ &= \frac{\log(2^n c_n)}{-\log(\delta)} - n + \frac{\log(N_\delta(F))}{-\log(\delta)}.\end{aligned}$$

As $\delta \to 0$, we have that $\frac{\log(2^n c_n)}{-\log(\delta)} \to 0$. Thus if we apply the lower and upper limit as $\delta \to 0$, we obtain

$$\varliminf_{\delta\to 0} \frac{\log(\mathcal{L}^n(F_\delta))}{-\log(\delta)} \leq -n + \underline{\dim}_{\mathrm{B}} F \quad \text{and} \quad \varlimsup_{\delta\to 0} \frac{\log(\mathcal{L}^n(F_\delta))}{-\log(\delta)} \leq -n + \overline{\dim}_{\mathrm{B}} F. \tag{7}$$

For the opposite inequality, let $N_\delta(F)$ be the largest number of balls of radius δ with centres in F, then they also have centres in F_δ, so that the volume of F_δ is at least the volume of these balls, so that

$$N_\delta(F)c_n\delta^n \leq \mathcal{L}^n(F_\delta).$$

Again, we take the logarithms and divide by $-\log(\delta)$ to obtain

$$\frac{\log(N_\delta(F))}{-\log(\delta)} + \frac{\log(c_n)}{-\log(\delta)} - n \leq \frac{\log(\mathcal{L}^n(F_\delta))}{\log(\delta)}.$$

Applying the lower and upper limits as $\delta \to 0$, we find

$$\underline{\dim}_{\mathrm{B}} F - n \leq \varliminf_{\delta\to 0} \frac{\log(\mathcal{L}^n(F_\delta))}{-\log(\delta)}, \quad \text{and} \quad \overline{\dim}_{\mathrm{B}} F - n \leq \varlimsup_{\delta\to 0} \frac{\log(\mathcal{L}^n(F_\delta))}{-\log(\delta)}. \tag{8}$$

We conclude by combining the inequalities in (7) and (8) to get the desired result. □

Remark 4.10. From this proposition follows that the box-counting dimension is often called the *Minkowski-Bouligand dimension*. In fact, if F and F_δ are as in the proposition and $\lim_{\delta\to 0} \frac{\mathcal{L}^n(F_\delta)}{\delta^{n-s}} = c$, then after some calculations, we check that s is the box-dimension of F. The name of the dimension then comes from the name of the constant c, that is called the *s-dimensional Minkowski content* of F.

[1]Since c_n is the volume of the unit ball in $\mathbb{R}^n$, $c_n \cdot (2\delta)^n$ is the volume of a ball of radius 2δ in $\mathbb{R}^n$. At least $N_\delta(F)$ such balls cover F_δ, that is why we multiply the obtained volume by $N_\delta(F)$ to get the volume of F_δ.

Proposition 4.11. Let E and F be two non-empty subsets of $\mathbb{R}^n$, then the (lower or upper) box counting dimension has the following properties.

(a) *Monotonicity*: If $E \subset F$, then $\underline{\dim}_{\mathrm{B}} E \leq \underline{\dim}_{\mathrm{B}} F$ and $\overline{\dim}_{\mathrm{B}} E \leq \overline{\dim}_{\mathrm{B}} F$.

(b) *Range of values*: If F is a non-empty bounded subset of $\mathbb{R}^n$, then

$$0 \leq \underline{\dim}_{\mathrm{B}} F \leq \overline{\dim}_{\mathrm{B}} F \leq n.$$

(c) *Finite stability*: $\overline{\dim}_{\mathrm{B}}$ is finitely stable, this means that

$$\overline{\dim}_{\mathrm{B}}(E \cup F) = \max\{\overline{\dim}_{\mathrm{B}} E, \overline{\dim}_{\mathrm{B}} F\}.$$

(d) *Open sets*: If $F \subset \mathbb{R}^n$ is open, then $\dim_B F = n$.

(e) *Finite sets*: If F is non-empty and finite, then $\dim_B F = 0$.

(f) *Smooth sets*: If F is a smooth (i.e. continuously differentiable) bounded m-dimensional surface of $\mathbb{R}^n$, then $\dim_B F = m$.
In particular, smooth curves have dimension 1 and smooth surfaces have dimension 2.

Proof. (a) If $E \subset F$, then clearly $N_\delta(E) \leq N_\delta(F)$, which implies that

$$\log(N_\delta(E)) \leq \log(N_\delta(F)).$$

We conclude, by dividing by $-\log(\delta)$ and taking the lower or upper limit, that $\underline{\dim}_{\mathrm{B}} E \leq \underline{\dim}_{\mathrm{B}} F$ and $\overline{\dim}_{\mathrm{B}} E \leq \overline{\dim}_{\mathrm{B}} F$.

(b) By definition, $\underline{\dim}_{\mathrm{B}} F \leq \overline{\dim}_{\mathrm{B}} F$ and $0 \leq \underline{\dim}_{\mathrm{B}} F$, since $N_\delta(F) \geq 1$ and $\underline{\lim}_{\delta \to 0} - \log(\delta) \geq 0$. This proves the first two inequalities.
In order to prove that $\overline{\dim}_{\mathrm{B}} F \leq n$, let us consider a large cube C that contains F. By covering F and C with δ-mesh cubes that intersect them, we see that $N_\delta(F) \leq N_\delta(C)$, because $F \subset C$.
Moreover, the volume of the δ-mesh that intersects C is finite, because F is bounded. Hence, this implies that the sides of C have a finite length. Thus, there exists some real constant $c > 0$ such that $N_\delta(C)\delta^n \leq c$.[2] Consequently, we have that $N_\delta(C) \leq c\delta^{-n}$. Thus,

$$\begin{aligned} & N_\delta(F) \leq c\delta^{-n} \\ \iff & \log(N_\delta(F)) \leq \log(c) - n\log(\delta) \\ \iff & -\frac{\log(N_\delta(F))}{\log(\delta)} \leq -\frac{\log(c)}{\log(\delta)} + n \end{aligned}$$

[2] $N_\delta(C)\delta^n$ is the volume of the δ-mesh that covers C.

Finally, taking the upper limit as $\delta \to 0$, we obtain that $\overline{\dim}_{\mathrm{B}} F \leq n$.

(c) First, we show that $\max\{\overline{\dim}_{\mathrm{B}} E, \overline{\dim}_{\mathrm{B}} F\} \leq \overline{\dim}_{\mathrm{B}}(E \cup F)$. Since $E \subset E \cup F$ and $F \subset E \cup F$, we know by monotonicity of the box dimension, that $\overline{\dim}_{\mathrm{B}} E \leq \overline{\dim}_{\mathrm{B}}(E \cup F)$ and $\overline{\dim}_{\mathrm{B}} F \leq \overline{\dim}_{\mathrm{B}}(E \cup F)$. Thus $\max\{\overline{\dim}_{\mathrm{B}} E, \overline{\dim}_{\mathrm{B}} F\} \leq \overline{\dim}_{\mathrm{B}}(E \cup F)$.

Let us now show that $\overline{\dim}_{\mathrm{B}}(E \cup F) \leq \max\{\overline{\dim}_{\mathrm{B}} E, \overline{\dim}_{\mathrm{B}} F\}$. It is easy to see that $N_\delta(E \cup F) \leq N_\delta(E) + N_\delta(F)$. As almost in every proof, we return to the definition by applying the logarithm and dividing by $-\log(\delta)$.
Finally, taking the upper limit gives $\overline{\dim}_{\mathrm{B}}(E \cup F) \leq \max\{\overline{\dim}_{\mathrm{B}} E, \overline{\dim}_{\mathrm{B}} F\}$, because the supremum of $\frac{\log(N_\delta(E)+N_\delta(F))}{-\log(\delta)}$ is $\max\{\overline{\dim}_{\mathrm{B}} E, \overline{\dim}_{\mathrm{B}} F\}$.

(d) From (b), we know that $\dim_B F \leq n$. In order to prove that $n \leq \dim_B F$ we consider a cube C that is contained in F. Covering F and C with δ-mesh cubes, we see that $N_\delta(F) \geq N_\delta(C)$. Then, since $N_\delta(C)$ cubes of side δ intersect C, the volume of this δ- mesh is larger than the volume of C, so that $N_\delta(C)\delta^n \geq c$, where $c \leq vol^n(C)$. This means that $N_\delta(C) \geq c\delta^{-n}$. Thus, we have the following inequality

$$N_\delta(F) \geq N_\delta(C) \geq c\delta^{-n}.$$

Finally, we take the logarithm and divide by $-\log(\delta)$. After taking the limit as $\delta \to 0$, we obtain $\dim_B F \geq n$. Combining the two inequalities, we get, as desired, $\dim_B F = n$.

(e) Let $m < \infty$ be the number of points in F. As δ tends to 0, the smallest number of sets of size δ, that cover F is exactly the number of elements of F. This implies that $\lim_{\delta \to 0} N_\delta(F) = m$. Thus

$$\dim_B F = \lim_{\delta \to 0} \frac{N_\delta(F)}{-\log(\delta)} = 0. \qquad \square$$

Remark 4.12. The identity in (c) is only true for a finite union of sets, furthermore, this identity does not hold for the lower limit, which means that $\underline{\dim}_{\mathrm{B}}(E \cup F) \neq \max\{\underline{\dim}_{\mathrm{B}} E, \underline{\dim}_{\mathrm{B}} F\}$.
In fact, for $\underline{\dim}_{\mathrm{B}}$ to be finitely stable, it must be satisfied that for two non-empty subsets E and F of $\mathbb{R}^n$, $\underline{\dim}_{\mathrm{B}}(E \cup F) = \max\{\underline{\dim}_{\mathrm{B}} E, \underline{\dim}_{\mathrm{B}} F\}$, which is not the case. With the same reasoning as in the proof of (c), we get by monotonicity, that $\max\{\underline{\dim}_{\mathrm{B}} E, \underline{\dim}_{\mathrm{B}} F\} \leq \underline{\dim}_{\mathrm{B}}(E \cup F)$. However, the opposite inequality does not always hold. In order to show this, we consider the following example.

Example 4.13. ([1], Exercise 2.9 p. 42.) Let $t_m := 10^m$ for every integer $m \in \mathbb{N}$ and consider $k \in \mathbb{N}$. We construct E and F similarly to the Cantor set, that is, we start with the unit interval and at every k^{th} stage of construction, we either delete the middle $\frac{1}{3}$ or the middle $\frac{3}{5}$ of the intervals. In the construction of E we delete

the middle third if $t_{2m} < k \leq t_{2m+1}$ and the middle $\frac{3}{5}$ if $t_{2m-1} < k \leq t_{2m}$. F is constructed by doing the opposite.
In conclusion, if $1 < k \leq 10$, then at the k^{th} stage of construction of E we delete the middle $\frac{1}{3}$ of the intervals and at that same stage of construction of F we delete the middle $\frac{3}{5}$ of the intervals. The construction continues infinitely often, so that for both sets, at each stage of construction there are 2^k intervals of length 3^{-k} or 5^{-k}.

It is easy to see that the union of E and F is simply the middle third Cantor set. This is because the union of the 2^k intervals of length 5^{-k}, that are left after deleting the middle $\frac{3}{5}$ part of the previous intervals, and the 2^k intervals of length 3^{-k}, that are left, after deleting the middle third part of the previous intervals, are the larger intervals, which are the 2^k intervals of length 3^{-k}. Thus, Example 4.5 gives us that

$$\underline{\dim}_{\mathrm{B}}(E \cup F) = \frac{\log(2)}{\log(3)}. \tag{9}$$

What is left to do now is to find the lower box dimension of F and E or an upper bound of this dimension. As already noted before, both sets consist of 2^k intervals of length 5^{-k} or 3^{-k} which cover them. If we chose δ such that $5^{-k} < \delta \leq 5^{-k+1}$, then, automatically, $3^{-k} < \delta$. Hence, every interval of length δ covers the intervals of length 3^{-k} and the intervals of length 5^{-k}, so that 2^k intervals of length δ provide a covering of F and of E. This implies that the smallest number of coverings is at most 2^k, so that $N_\delta(F) \leq 2^k$ and $N_\delta(E) \leq 2^k$.
By Definition 4.1, we obtain

$$\underline{\dim}_{\mathrm{B}} E = \underline{\dim}_{\mathrm{B}} F = \underline{\lim}_{\delta \to 0} \frac{\log(N_\delta(F))}{-\log(\delta)} \leq \underline{\lim}_{k \to \infty} \frac{\log(2^k)}{-\log(5^{-k+1})} = \frac{\log(2)}{\log(5)}. \tag{10}$$

Finally, combining (9) and (10) we get that

$$\max\{\underline{\dim}_{\mathrm{B}} E, \underline{\dim}_{\mathrm{B}} F\} \leq \frac{\log(2)}{\log(5)} < \frac{\log(2)}{\log(3)} = \underline{\dim}_{\mathrm{B}}(E \cup F).$$

Hence, we conclude that $\underline{\dim}_{\mathrm{B}}$ is not finitely stable.

Proposition 4.14. Let $m \in \mathbb{N}$. Consider the non-empty subset $F \subset \mathbb{R}^n$ and the function $f : F \to \mathbb{R}^m$.

(i) If, in addition, F is compact, $\alpha > 0$, and f is a α-Hölder map, then $\underline{\dim}_{\mathrm{B}} f(F) \leq \frac{1}{\alpha} \underline{\dim}_{\mathrm{B}} F$ and $\overline{\dim}_{\mathrm{B}} f(F) \leq \frac{1}{\alpha} \overline{\dim}_{\mathrm{B}} F$

(ii) If f is a Lipschitz transformation, then $\underline{\dim}_{\mathrm{B}} f(F) \leq \underline{\dim}_{\mathrm{B}} F$ and $\overline{\dim}_{\mathrm{B}} f(F) \leq \overline{\dim}_{\mathrm{B}} F$.

(iii) If f is a bi-Lipschitz transformation, then $\underline{\dim}_{\mathrm{B}} f(F) = \underline{\dim}_{\mathrm{B}} F$ and $\overline{\dim}_{\mathrm{B}} f(F) = \overline{\dim}_{\mathrm{B}} F$.

Proof. (i) Let $\{U_i\}_{i\geq 1}$ be a δ-cover of F, then since $F \subset \bigcup_{i=1}^{\infty} U_i \cap F$ and $|U_i \cap F| \leq |U_i| \leq \delta$, $U_i \cap F$ is also a δ-cover of F. Taking the image over f, we have $f(F) \subset \bigcup_{i=1}^{\infty} f(U_i \cap F)$ and, by α-Hölder-continuity of f, there exists a constant $c > 0$ such that

$$|f(U_i \cap F)| \leq c|U_i \cap F|^{\alpha} \leq c|U_i|^{\alpha} \leq c\delta^{\alpha}.$$

Hence $\{f(U_i \cap F)\}$ is a $c\delta^{\alpha}$-cover of $f(F)$. This implies that every δ-cover of F forms a $c\delta^{\alpha}$-cover of $f(F)$.
Now, let $N_{\delta}(F)$ be the smallest number of δ-covers of F, then there are at least $N_{\delta}(F)$ $c\delta^{\alpha}$-covers of $f(F)$. This implies that $N_{c\delta^{\alpha}}(f(F)) \leq N_{\delta}(F)$. Applying the logarithm and dividing by $-\alpha\log(\delta)$, we obtain

$$\frac{\log(N_{c\delta^{\alpha}}(f(F)))}{-\alpha\log(\delta)} \leq \frac{\log(N_{\delta}(F))}{-\alpha\log(\delta)}.$$

Calculations on the left hand side then, give us

$$\frac{\log(N_{c\delta^{\alpha}}(f(F)))}{-\log(c\delta^{\alpha}) + \log(c)} \leq \frac{1}{\alpha}\frac{\log(N_{\delta}(F))}{-\log(\delta)}.$$

Finally, taking the lower or upper limit as $\delta \to 0$, we find as desired $\underline{\dim}_{\mathrm{B}} f(F) \leq \frac{1}{\alpha}\underline{\dim}_{\mathrm{B}} F$ and $\overline{\dim}_{\mathrm{B}} f(F) \leq \frac{1}{\alpha}\overline{\dim}_{\mathrm{B}} F$.

(ii) This follows directly from (i) for $\alpha = 1$.

(iii) Since f is a bi-Lipschitz transformation, it is also a Lipschitz transformation. Thus by (i), we obtain that $\underline{\dim}_{\mathrm{B}} f(F) \leq \underline{\dim}_{\mathrm{B}} F$ and $\overline{\dim}_{\mathrm{B}} f(F) \leq \overline{\dim}_{\mathrm{B}} F$.

Let us now show the opposite inequalities. We know that if f is bi-Lipschitz, then $f : F \to f(F) \subset \mathbb{R}^m$ is bijective with inverse $f^{-1} : f(F) \to F$. Let $u, v \in f(F)$, such that $x := f^{-1}(u)$ and $y := f^{-1}(v)$. Then, by bi-Lipschitz continuity, and choosing $c_1 \in \mathbb{R}_+$, we obtain

$$c_1|f^{-1}(u) - f^{-1}(v)| = c_1|x - y| \leq |f(x) - f(y)| = |f(f^{-1}(u)) - f(f^{-1}(v))| = |u - v|.$$

Hence, f^{-1} is a Lipschitz transformation. Applying (i) to f^{-1}, we obtain that $\underline{\dim}_{\mathrm{B}} F = \underline{\dim}_{\mathrm{B}} f^{-1}(f(F)) \leq \underline{\dim}_{\mathrm{B}} f(F)$ and $\overline{\dim}_{\mathrm{B}} F = \overline{\dim}_{\mathrm{B}} f^{-1}(f(F)) \leq \overline{\dim}_{\mathrm{B}} f(F)$. Finally, by combining the inequalities, we can conclude that $\underline{\dim}_{\mathrm{B}} f(F) = \underline{\dim}_{\mathrm{B}} F$ and $\overline{\dim}_{\mathrm{B}} f(F) = \overline{\dim}_{\mathrm{B}} F$. □

From Proposition 4.14 some properties follow.

Proposition 4.15. (a) *Geometric invariance*: For $m \in \mathbb{N}$, if $f : F \to \mathbb{R}^m$ is a congruence, similarity or affine transformation, then $\underline{\dim}_\mathrm{B} f(F) = \underline{\dim}_\mathrm{B} F$ and $\overline{\dim}_\mathrm{B} f(F) = \overline{\dim}_\mathrm{B} F$.

(b) *Smooth curves*: If $g : [0, 1] \to \mathbb{R}$ is a Lipschitz transformation and $\Gamma(g) = \{(x, g(x)) : x \in [0, 1]\}$ is the graph of the function g, then $\dim_B \Gamma(g) = 1$.

(c) *Projections*: If F is a subset of $\mathbb{R}^2$, then $\underline{\dim}_\mathrm{B}\, proj(F) \le \min\{1, \underline{\dim}_\mathrm{B} F\}$ and $\overline{\dim}_\mathrm{B}\, proj(F) \le \min\{1, \overline{\dim}_\mathrm{B} F\}$, where *proj* denotes the orthogonal projection from $\mathbb{R}^2$ onto some given line through the origin.

Proof. (a) Since congruences, similarities and affine transformations are all bi-Lipschitz, then by Proposition 4.14 the result follows immediately.

(b) Consider $f : [0, 1] \to \Gamma(g)$ given by $f(x) = (x, g(x))$. Notice that f is bi-Lipschitz and $f([0, 1]) = \Gamma(g)$. In fact, for all $x, y \in [0, 1]$

$$|f(x) - f(y)| = |(x, g(x)) - (y, g(y))| = \sqrt{|x-y|^2 + |g(x) - g(y)|^2}.$$

Since $|g(x) - g(y)|^2 \ge 0$, it is clear that $|x - y| \le |f(x) - f(y)|$. Moreover, g is a Lipschitz transformation, which implies that

$$\sqrt{|x-y|^2 + |g(x)-g(y)|^2} \le \sqrt{|x-y|^2 + c^2|x-y|^2} = \sqrt{1+c^2}|x-y|, \quad \forall x, y,$$

for some positive constant c. This implies that $|f(x)-f(y)| \le \sqrt{1+c^2}|x-y|$. Hence f is bi-Lipschitz continuous. From Proposition 4.14, it follows that $\dim_B \Gamma(g) = \dim_B f(F) = \dim_B[0, 1]$. Since intervals are smooth 1-dimensional surfaces, we conclude that $\dim_B \Gamma(g) = \dim_B[0, 1] = 1$.

(c) It is easy to check that orthogonal projections do not increase distances, thus

$$|proj(x) - proj(y)| \le |x - y| \quad \text{for } x,\, y \in \mathbb{R}.$$

Hence, we conclude that *proj* is a Lipschitz transformation. By Proposition 4.14 $\underline{\dim}_\mathrm{B}\, proj(F) \le \min\{1, \underline{\dim}_\mathrm{B} F\}$ and $\overline{\dim}_\mathrm{B}\, proj(F) \le \min\{1, \overline{\dim}_\mathrm{B} F\}$. □

Proposition 4.16. Let $\overline{F}$ denote the closure of F, i.e. the smallest closed subset of $\mathbb{R}^n$ containing F. Then

$$\underline{\dim}_\mathrm{B} \overline{F} = \underline{\dim}_\mathrm{B} F \quad \text{and} \quad \overline{\dim}_\mathrm{B} \overline{F} = \overline{\dim}_\mathrm{B} F.$$

Proof. For $k \in \mathbb{N}$, let $\{B_1, B_2, ...B_k\}$ be a finite collection of balls of radius δ in $\mathbb{R}^n$. The union of all these balls is a closed set. Thus, since $\overline{F}$ is the smallest closed set containing F, these balls form a δ-cover of F if and only if they also form a δ-cover of $\overline{F}$. Hence, we conclude that $N_\delta(F) = N_\delta(\overline{F})$, which implies that $\underline{\dim}_\mathrm{B} \overline{F} = \underline{\dim}_\mathrm{B} F$ and $\overline{\dim}_\mathrm{B} \overline{F} = \overline{\dim}_\mathrm{B} F$. □

Example 4.17. Let us show that, if $F = \{0,\, 1,\, \frac{1}{2},\, \frac{1}{3},\, ...\}$ is a compact subset of $\mathbb{R}$, then $\dim_B F = \frac{1}{2}$. To do this, we will show that $\underline{\dim}_{\mathrm{B}} F \geq \frac{1}{2}$ and $\overline{\dim}_{\mathrm{B}} F \leq \frac{1}{2}$, then

$$\frac{1}{2} \leq \underline{\dim}_{\mathrm{B}} F \leq \overline{\dim}_{\mathrm{B}} F \leq \frac{1}{2},$$

hence $\overline{\dim}_{\mathrm{B}} F = \underline{\dim}_{\mathrm{B}} F = \dim_B F = \frac{1}{2}$.

Let us start by proving that $\underline{\dim}_{\mathrm{B}} F \geq \frac{1}{2}$. Let $0 < \delta < \frac{1}{2}$ and $k \in \mathbb{N}^*$, so that $\frac{1}{(k-1)k} > \delta \geq \frac{1}{k(k+1)}$.
The distance between each couple of points in $\{1,\, \frac{1}{2},\, ...\, \frac{1}{k}\}$ is at least $\frac{1}{k-1} - \frac{1}{k} = \frac{1}{(k-1)k} > \delta$. This implies that any set U of diameter at most δ, covers at most one of the points $\{1,\, \frac{1}{2},\, ...\, \frac{1}{k}\}$. Hence, to cover F, we need at least k sets of diameter at most δ, which implies that $N_\delta(F) \geq k$.
Moreover, if $\delta \to 0$, then $k \to \infty$, because $\frac{1}{(k-1)k} > \delta \geq \frac{1}{k(k+1)}$.

By Definition 4.1 (a), we obtain

$$\begin{aligned}\underline{\dim}_{\mathrm{B}} F = \underline{\lim}_{\delta\to 0} \frac{\log(N_\delta(F))}{-\log(\delta)} &\geq \underline{\lim}_{k\to\infty} \frac{\log(k)}{\log(k(k+1))} \\ &= \underline{\lim}_{k\to\infty} \frac{\log(k)}{\log(k^2(1+\frac{1}{k}))} \\ &= \underline{\lim}_{k\to\infty} \frac{\log(k)}{2\log(k) + \log(1+\frac{1}{k})} \\ &= \frac{1}{2}.\end{aligned}$$

Let us now show that $\overline{\dim}_{\mathrm{B}} F \leq \frac{1}{2}$. We choose δ as above and this time, we choose $k \in \mathbb{N}^*$ so that $\frac{1}{(k-1)k} \geq \delta > \frac{1}{k(k+1)}$. We see that $k+1$ intervals of length δ cover the interval $[0,\, \frac{1}{k}]$. However, this leaves out $k-1$ points of F (the ones in $]\frac{1}{k},\, 1]$), that we cover with another $k-1$ intervals. Hence, we found $k+1+k-1 = 2k$ intervals that cover F. This implies that $N_\delta(F) \leq 2k$, thus by Definition 4.1 (b) and by similar calculations as above, we get

$$\overline{\dim}_{\mathrm{B}} F = \overline{\lim}_{\delta\to 0} \frac{\log(N_\delta(F))}{-\log(\delta)} \leq \frac{1}{2}.$$

We conclude, as desired, that $\frac{1}{2} \leq \underline{\dim}_{\mathrm{B}} F \leq \overline{\dim}_{\mathrm{B}} F \leq \frac{1}{2}$, thus $\dim_B F = \frac{1}{2}$.

Example 4.18. Let us calculate the box-counting dimension of $F = \left\{0, 1, \frac{1}{4}, \frac{1}{9}, ...\right\}$.

We start by giving an upper bound for the lower box-dimension. Let $\delta > 0$ and

$k \in \mathbb{N}$ such that $\frac{1}{(k-1)^2k^2} > \delta \geq \frac{1}{k^2(k+1)^2}$.
The distance between each couple of points in $\left\{1, \frac{1}{4}, \frac{1}{9}, ..., \frac{1}{k^2}\right\}$ is at least $\frac{1}{(k-1)^2} - \frac{1}{k^2} = \frac{2k-1}{(k-1)^2k^2} \geq \frac{1}{(k-1)^2k^2} > \delta$. This implies that every set U of diameter at most δ, covers at most one of the points in $\left\{1, \frac{1}{4}, \frac{1}{9}, ..., \frac{1}{k^2}\right\}$.
Hence, to cover F, we need at least k sets of diameter at most δ, which implies that $N_\delta(F) \geq k$.

By Definition 4.1 (a), we obtain

$$\begin{aligned}\underline{\dim}_{\mathrm{B}} F = \underline{\lim}_{\delta \to 0} \frac{\log(N_\delta(F))}{-\log(\delta)} &\geq \underline{\lim}_{k \to \infty} \frac{\log(k)}{\log(k^2(k+1)^2)} \\ &= \underline{\lim}_{k \to \infty} \frac{\log(k)}{\log(k^4(1+\frac{1}{k})^2)} \\ &= \underline{\lim}_{k \to \infty} \frac{\log(k)}{4\log(k) + 2\log(1+\frac{1}{k})} \\ &= \frac{1}{4}.\end{aligned}$$

Let us now show that the upper bound of the lower box-dimension is the same as the lower bound of the upper box-dimension. We choose δ as above and $k \in \mathbb{N}^*$ so that $\frac{1}{(k-1)^2k^2} \geq \delta > \frac{1}{k^2(k+1)^2}$. With similar reasoning as in the previous example, we find $k+1+k-1 = 2k$ intervals that cover F. This implies that $N_\delta(F) \leq 2k$, so that by Definition 4.1 (b), and by similar calculations as above,

$$\overline{\dim}_{\mathrm{B}} F = \overline{\lim}_{\delta \to 0} \frac{\log(N_\delta(F))}{-\log(\delta)} \leq \frac{1}{4}.$$

Thus, $\overline{\dim}_{\mathrm{B}} F \leq \frac{1}{4} \leq \underline{\dim}_{\mathrm{B}} F$. By definition, $\overline{\dim}_{\mathrm{B}} F \geq \underline{\dim}_{\mathrm{B}} F$, hence we obtain

$$\overline{\dim}_{\mathrm{B}} F = \underline{\dim}_{\mathrm{B}} F = \dim_B F = \frac{1}{4}.$$

4.2 Comparing the Hausdorff and the box-counting dimension

After having examined the box-counting dimension and also looked at the Hausdorff dimension, we notice that they have similar properties, that is why in this subsection, we will compare the two dimensions.

Proposition 4.19. For every non-empty bounded $F \subset \mathbb{R}^n$

$$\dim_H F \leq \underline{\dim}_{\mathrm{B}} F \leq \overline{\dim}_{\mathrm{B}} F.$$

Proof. The second inequality follows from the Definition 4.1, therefore it suffices to prove the first inequality.
Let $s \geq 0$, such that $1 \leq \mathcal{H}^s(F) = \lim_{\delta \to 0} \mathcal{H}^s_\delta(F)$. By Definition 3.1, of the s-dimensional Hausdorff measure, we have that $1 \leq \lim_{\delta \to 0} \mathcal{H}^s_\delta(F) \leq \delta^s N_\delta(F)$, where $N_\delta(F)$ represents the smallest number of δ-covers of F. Hence, applying the logarithm gives us $0 \leq \log(\delta^s N_\delta(F)) = s\log(\delta) + \log(N_\delta(F))$. This implies that $-\log(\delta)s \leq \log(N_\delta(F))$. If we take the lower limit as δ goes to 0 and divide by $-\log(\delta)$, we get

$$s \leq \underline{\lim}_{\delta \to 0} \frac{\log(N_\delta(F))}{-\log(\delta)}.$$

Since this is true for every $s \geq 0$ such that $\mathcal{H}^s(F) > 1$, it is also true for $\dim_H F = \sup\{s \geq 0 : \mathcal{H}^s(F) = \infty\}$. Thus we conclude that $\dim_H F \leq \underline{\dim}_{\mathrm{B}} F$. $\square$

In general, the inequality between the Hausdorff and the box-counting dimension is strict, however there are some cases for which the dimensions are equal. One example is the middle third Cantor set. In Example 3.7, we found $\dim_H F = \frac{\log(2)}{\log(3)}$ and in Example 4.5, we found $\dim_B F = \underline{\dim}_{\mathrm{B}} F = \overline{\dim}_{\mathrm{B}} F = \frac{\log(2)}{\log(3)}$.

Let us now look at the properties they have or have not in common.
One common property is that they are both invariant under bi-Lipschitz transformations, see Propositions 3.5 and 4.14.
Also, if we compare Propositions 3.4 and 3.6 to Propositions 4.11 and 4.15, we see that both dimensions have a lot of common properties, listed below.

- Monotonicity, geometrical invariance and they have the same range of values.
- If E is an open non-empty subset of $\mathbb{R}^n$, then both dimensions of E are exactly n.
- If we consider smooth m-dimensional sets or curves, then the Hausdorff and the box-dimension are both equal to $m \in \mathbb{N}$.
- The box and the Hausdorff dimension of a projection of a non-empty subset of $\mathbb{R}^n$ is smaller than the dimension of the subset itself.

However, they also have different properties, for instance, the upper box-dimension is finitely stable, whereas the Hausdorff dimension is countably stable. Also, the box dimension of finite sets is 0, whereas for the Hausdorff dimension, the countable sets that have 0 dimension.

One of the advantages of the box-counting dimension compared to a lot of other fractal dimension is that it is easier to calculate. Mainly the box-counting version of the definition is very often used for experimentations, as we will see in the following section.

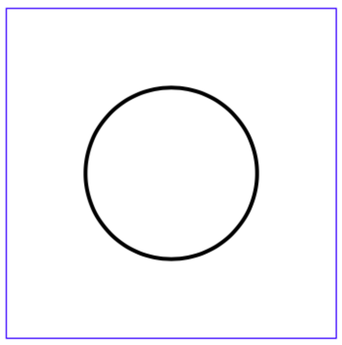
5 Box-counting method

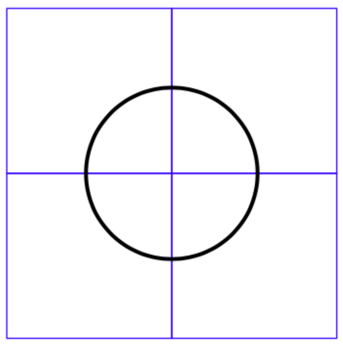
The box-counting method is an approach used to calculate the fractal dimension of different surfaces F, for example a coastline or the Sierpiński triangle. The first step of the box-counting method is to draw a grid over the figure you want to measure and count the number of boxes that contain parts of the given figure. Next, we repeat this procedure for different sizes of the boxes in the grid. The last step in the box-counting method gives an approximation of the the box-dimension. In order to do this, we consider (3) and apply the logarithms to obtain

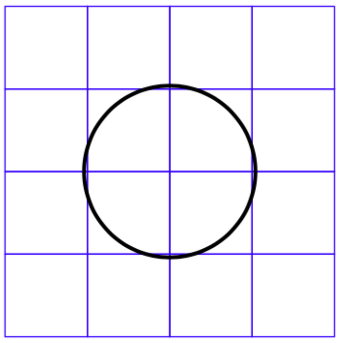
$$\log(N_\delta(F)) \simeq \log(c) - s\log(\delta),$$

where $N_\delta(F)$ denotes the number of boxes of side-length δ that intersect F, c is a positive constant, $\delta > 0$ is a real number and $s \geq 0$ denotes the box-counting dimension. We see that this is an equation of the form $y = mx + c$, i.e. of a straight line, where $y = \log(N_\delta(F))$, $x = \log(\delta)$ and $m = -s$. Thus, in order to find the box dimension, we need to find a linear regression of $\log(N_\delta(F))$ on $\log(\delta)$ and the opposite of the slope will be the box-counting dimension.
In order to understand how this method works, let us illustrate it by the following example.

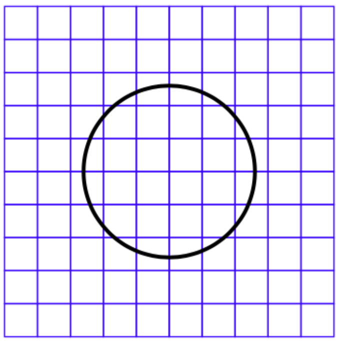
Example 5.1. Let us calculate the box-count dimension of a circle using the box counting method. We start by drawing grids of different scales over the circle.


(a)

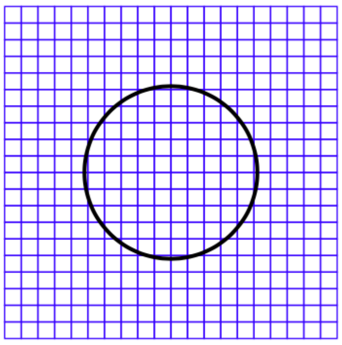

(b)



(c)



(d)



(e)

Figure 8: (a) A circle with a grid consisting of 1 square. (b) We split the squares from the previous grid in half. (c) We split the previous squares again in half, so that the squares have a length of $\frac{1}{4}$. (d) We draw a grid consisting of squares of length $\frac{1}{10}$. (e) We consider a grid with squares of side length $\frac{1}{20}$.

Counting the number of squares that intersect the line of the circle in each grid gives the following table

Scale	1	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{10}$	$\frac{1}{20}$
Number of boxes	1	4	12	20	44

Table 1: The number of boxes that intersect the circle in each figure above.

Now, applying the logarithms gives the following

$\log(\delta)$	0	$-0,6931$	$-1,3862$	$-2,3026$	$-2,9957$
$\log(N_\delta)$	0	$1,3862$	$2,4849$	$2,9957$	$3,7841$

Table 2: Applying the logarithms in the previous table.

Finally, we use *Sage*, to obtain that the polynomial regression is given by

$$\log(N_\delta(F)) = -1,1997 \cdot \log(\delta) + 3,9660,$$

so that $s = 1,1997$. However, since a circle is a smooth one-dimensional curve, the box-dimension should be 1. Clearly, in order to have a more accurate result, we should have chosen more grids, where the squares have a smaller size. However this is already a good approximation, considering the large scales we used.

The *Sage Code* used to visualise the grids is the following

```
import numpy as np
c = circle((0,0),5.2 , edgecolor = 'black', thickness = 3 , axes =
   False)                                  #plot circle
grid = Graphics()
k=1                         #choose k, the side-length of the squares
for i in range(-10,10,k):                     #start constructing the grid
    for j in range(-10,10,k):
        grid += polygon2d([(i,j),(i,j+k),(i+k,j+k),(i+k,j)], fill=
   False)
c + grid
```

Listing 1: Code used to plot the circle and the grid.

We start by plotting a circle with centre $(0,0)$ and radius 5,2. Then, we construct the grid by starting with an empty graphic that we extend with squares of side length k, where k is a randomly chosen integer.

After having counted the number of squares intersecting the circle by hand, we use another algorithm that allows us to find the linear regression. First, we create a list with the scales and a list with the number of squares, then compute the linear regression of the logarithm of the number of squares on the logarithm of the scales using the following code

```
import numpy as np
scales=[1 , 1/2 , 1/4 , 1/10 , 1/20 ]  #scales, length of the
   squares of the grids
N=[1 , 4 , 12 , 20 , 44]               #number of squares that were
    counted by hand
poly=np.polyfit(np.log(scales),np.log(N),1) #given by log(N) =
   Polyfit[0]*log(scales)**1 + Polyfit[1]
print('The fitting polynomial of N is:', poly[0],'x + ', poly[1]')
```

Listing 2: Code used to compute the linear regression.

5.1 Box-counting algorithm that uses the data of the image

Calculating the box dimension of a given figure, using the box-counting method can be pretty exhausting. This is because, in order to get a good approximation we have to consider a lot of different grids, so that we have to count a lot of squares, which is tiring if we do it by hand. For this reason, we try to write an algorithm that calculates the box dimension. The following *Sage Code*, that is given from [10], allows us to do so.

```
import cv2 as cv
import numpy as np
import pylab as pl


###################### First step: including picture and turn it grey
def rgb2grey(rgb):
    r, g, b = rgb[:,:,0], rgb[:,:,1], rgb[:,:,2]
    grey = 0.2989 * r + 0.5870 * g + 0.1140 * b      #mix colours to create grey
    return grey

image=rgb2grey(pl.imread("name.png")) #chose the picture with the figure we want to calculate the box dimension

###################### Second step: Detect and count the pixels that are not zero
pixels=[]
for i in range(image.shape[0]):          #go over the height of the image
    for j in range(image.shape[1]):      #go over the length of the image
        if image[i,j] > 0:               #check if image is white
            pixels.append((i,j))

Lx = image.shape[1]
Ly = image.shape[0]
pixel=pl.array(pixels)

###################### Third step: Give the scales and count the number of boxes that cover the image
scales=np.logspace(-2, 5.5,num=20, endpoint= False, base=np.exp(1))
N=[]
for scale in scales:                                     #go over each pixel
    H, edges = np.histogramdd(pixel, bins=(np.arange(0,Lx,scale),np.arange(0,Ly,scale)))    #analyse data of the image
    N.append(np.sum(H>0))

```

```
32 table([(scales[i],N[i]) for i in [0..len(N)-1]], frame = True,
       header_row=['Scale','N'])      #create table
33
34 ###################### Fourth step: Give a linear fit for the
       number of boxes
35 Polyfit = np.polyfit(np.log(scales),np.log(N),1)
36 print('\n The fitting Polynomial of N is:', Polyfit[0],'x + ',
       Polyfit[1],'\n')
37
38 ###################### Fifth step: Plot the linear fit and the
       number of boxes
39 pl.plot(np.log(scales),np.log(N), 'o', mfc='none')
40 pl.plot(np.log(scales), np.polyval(Polyfit,np.log(scales)))
41 pl.xlabel('log $\\delta$')
42 pl.ylabel('log N')
43 pl.savefig('Graph.pdf')
44
45 ###################### Last step: determine dimension
46 print('\n The Box dimension is given by', -Polyfit[0])
```

Listing 3: Algorithm that gives the box dimension of a figure by analysing its data in a histogram.

The above algorithm consists of six steps:

Step 1: **(Lines 6-13).** The goal of the first step is to include the picture, that contains the figure of which we want to know the dimension, and turn it grey. In order to do this, we start by defining a function named `rgb2grey`, which takes as a variable the array `rgb`, describing the colours of the given image. This function starts by assigning `r` to `rgb[:,:,0]`, the red pixels of the image, and similarly, it assigns `g` to the green pixels and `b` to the blue pixels of the image. Then, it assigns to `grey`, the colour that is obtained by mixing a certain amount of red, green and blue together. This function then returns the colour grey, this means that the array describing colours introduced in the function is now an array describing grey-scales.

Then, in line 12, we are going to apply the above defined function to the image of the figure of which we want to calculate the dimension. We start by "reading" the picture, which is done with the function `pl.imread('name of the image.png')`, to obtain an array containing every information of the image. The result of the application of the function to the inserted picture is called `image`.

Step 2: **(Lines 14-24).** After turning the colours into grey-scales, we come to the second step, where we create a list containing the position of every non-white pixel. In order to do this, we start with an empty list, named `pixels` to which we add items during a `for`-loop. In this loop, we have a variable `i`, that goes from 0 to the end of the height of the picture, which is given by `image.shape[0]`. In order to go over every pixel in the whole image, we also need to go over the length of the image, that is why we create another `for`-loop with the variable `j`, that starts at 0 and ends at the end of the length of the picture, which is given by `image.shape[1]`. Then, in this double-loop, we check if the pixels are white or not, by considering the case `if image[i,j]>0`. Here, `image[i,j]` is the pixel at height `i` and length `j` and if it is positive, then it is not white. Hence, if the above case is true, we add the point `(i,j)` to the list of pixels that are not white, by writing `pixels.append((i,j))`.

In line 21 and 22, we name `Lx` the length of the image and `Ly` the height of the image. Lastly, for this step, in line 23, we transform the list of couples containing the positions of the non-white pixels into an array, so that we can do operations with these pixels and the image, which is also given by an array.

Step 3: **(Lines 25-33).**This is the most important step, because here, we consider, illustratively speaking, boxes of different sizes and count how many of them contain grey pixels. We start by giving a list with the scales, that represent the "side-length" of the different "boxes"; in Section 4 denoted by δ.

We notice that for a better result, we need as many scales as possible, that is why we consider a logarithmic scale, which allows us to display a large number of numerical data. In *Python*, `np.logspace(start, stop, num, endpoint, base)` returns numbers that are evenly spaced in a logarithmic scale. Here, we use this function, with a start at `-2`, end at `5.5`, containing `20` values and the base is `np.exp(1)`, which is the exponential e. The list `scales`, given in line 26, contains the returned values of this function, which are given by e^k, for all the 20 k's between -2 and 5.5.

In line 27, we create an empty list `N`, which will later be completed with, "the number of boxes intersecting the image".

The main part of this step, is the `for`-loop that goes from line 28 to line 30. The variable of this loop is `scale`, which goes over all the values in the list `scales`, created in line 26. Next, we will need to "draw boxes of length `scale`" and count how many of them "contain non-white pixels".

In order to do this, let us recall that a multidimensional histogram allows us to look at the frequencies of given data by grouping it into bins ("classes" or "buckets") of equal width. This grouping creates a grid of bins over the data and these bins represent the "boxes" of length δ of the grid that we "draw" on the image. In *Python*, such a multidimensional histogram is obtained from `np.histogramdd(sample, bins)`, where `samples` corresponds to the data that needs to be analysed and `bins` is the sequence containing the arrays that describe the bin edges for each dimension of the data. Here, we want to analyse the pixels of the image, therefore the `sample` will be the array `pixel`. Since the image is two-dimensional (it has a length and a height), `bins` will be a sequence containing two arrays, one that describes the edges along the length of the image and one that will describe the edges along the height of the image. In *Python*, `np.arange(start, stop, step)` returns an array with values that are evenly spaced with a distance of `step`. The array that we want is one with the values that go from 0 to the end of the length `Lx` (resp. height `Ly`) of the image with a distance of `scale` between them, therefore we use the array `np.arrange(0, Lx, scale)` (resp. `np.arrange(0, Ly, scale)`). The variable `bins` will be the sequence containing these two arrays.

As we have chosen the variables of the histogram, we can evaluate the data of the image by executing in line 29 `H, edges = np.histogramdd(pixel, bins=(np.arange(0, Lx, scale), np.arange(0, Ly, scale)))`.

For `H`, this will return the data contained in `bins`, i.e. the positions of the non-white pixels that are in the bins. If there is no data in a given bin, then `H` will be 0, therefore, in order to get the number of bins that contain data, we sum every non-empty `H`, and we add it to the previously created list `N`, by writing `N.append(np.sum(H>0))`.

When the `for`-loop is finished, `N` will be a list containing the results of the sums of all the non-zero histograms. That is, the elements of `N` will be the number of bins of different sizes that contain data.

In order to have a better visualisation, we create a table in line 32, illustrating the items of `scales` and the items of `N`, so that we see how many bins contain data for the different scales.

Step 4: **(Lines 34-37).** The goal of this step is to find the linear fit for the logarithms of the items in `N` compared to the logarithms of the corresponding scales. In order to do this, we approximate the points that have as y-coordinate the logarithm of an item of `N` and as x-coordinate the logarithm of the corresponding scale, by a polynomial of degree 1. In *Python*, `Polyfit=np.polyfit(np.log(scales), np.log(N),1)` returns a list with the coefficients of such a polynomial. In line 37, we simply print a sentence explaining what the equation of this polynomial looks like.

Step 5: **(Lines 38-44).** This step simply consists of plotting all the points that have as coordinates the logarithms of the items of `N` and the logarithms of the scales, as well as the line that approximates these points. In line 41, we plot the points, in line 42 the line and in line 43 and 44 we add descriptions to the axes.

Step 6: **(Lines 45-46).** Finally, in the **last step**, we just print a sentence saying what the box-dimension is.

This concludes the explanation of the algorithm, so let us now look at its outputs.

5.1.1 Outputs

The goal of this thesis is to find the box-dimension of the borders of Luxembourg and we hope that the algorithm of this section allows us to determine it. In order to check that the obtained result will be correct, we will test the algorithm for figures of which we know the fractal dimension.

Example 5.2 (Testing the Algorithm for a circle and explaining the output)**.** In Proposition 4.11 (f), we learned that the box-dimension of a one-dimensional curve is 1, so that the dimension of a circle is 1. Let us check if the algorithm gives the same result.
We choose the following picture, that was created with *Geogebra*.

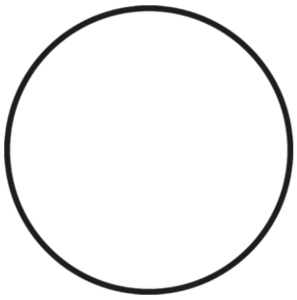
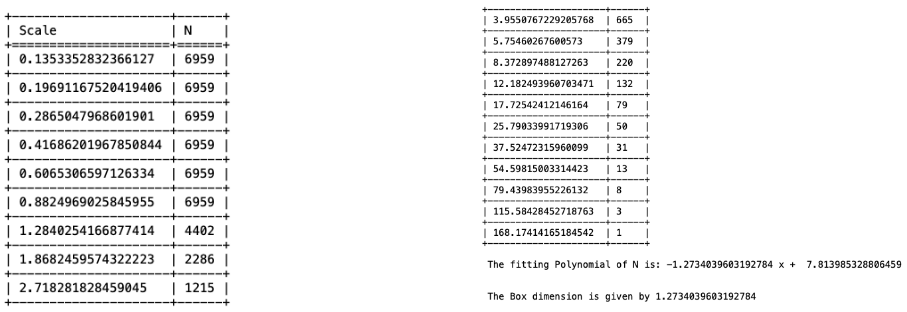


Figure 9: Image of a circle.

Running the algorithm with this picture, we obtain the following output.



```
+---------------------+------+
| Scale               | N    |
+=====================+======+
| 0.1353352832366127  | 6959 |
+---------------------+------+
| 0.19691167520419406 | 6959 |
+---------------------+------+
| 0.2865047968601901  | 6959 |
+---------------------+------+
| 0.41686201967850844 | 6959 |
+---------------------+------+
| 0.6065306597126334  | 6959 |
+---------------------+------+
| 0.8824969025845955  | 6959 |
+---------------------+------+
| 1.2840254166877414  | 4402 |
+---------------------+------+
| 1.8682459574322223  | 2286 |
+---------------------+------+
| 2.718281828459045   | 1215 |
+---------------------+------+
```

```
+---------------------+------+
| 3.9550767229205768  | 665  |
+---------------------+------+
| 5.75460267600573    | 379  |
+---------------------+------+
| 8.372897488127263   | 220  |
+---------------------+------+
| 12.182493960703471  | 132  |
+---------------------+------+
| 17.72542412146164   | 79   |
+---------------------+------+
| 25.79033991719306   | 50   |
+---------------------+------+
| 37.52472315960099   | 31   |
+---------------------+------+
| 54.59815003314423   | 13   |
+---------------------+------+
| 79.43983955226132   | 8    |
+---------------------+------+
| 115.58428452718763  | 3    |
+---------------------+------+
| 168.17414165184542  | 1    |
+---------------------+------+

The fitting Polynomial of N is: -1.2734039603192784 x +  7.813985328806459

The Box dimension is given by 1.2734039603192784
```

Figure 10: Output of the algorithm for a circle.

The output of the algorithm is a table containing the scales and the number N of bins containing some data, followed by two sentences, one giving the fitting polynomial and the other one giving the box dimension. In the table, we observe that the first six values of N are the same, and they decrease from the seventh value to the end. The most important part of the output is the obtained box dimension, which is in this example more or less 1.27.

Let us now look at the result obtained for another curve of which we know the dimension.

Example 5.3. As said in the introduction, Lewis Fry Richardson found out that coastline of Great Britain has a fractal dimension of 1.25, so let us check what value the algorithm gives us. We choose the following picture of the coastline of Britain.

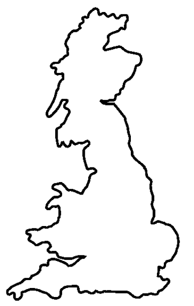


Figure 11: The coastline of Great Britain, [11].

For the coastline of Great Britain, the algorithm returns a box-dimension of approximately 1.61.

Remark 5.4. We notice that in both examples, the obtained approximation of the box dimension is not very precise. In the first example, we get an error of 0.27 and for Britain, the error is 0.36. There are several reasons for this "bad" approximation, that we will discuss in the following subsection.

Even if the algorithm is not very precise, we will still use it to get an approximation of the box dimension of the borders of Luxembourg.

Example 5.5. Let us choose the following picture of Luxembourg.

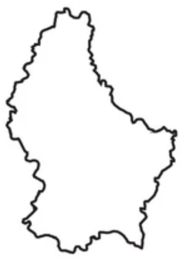


Figure 12: The borders of Luxembourg, [12].

The obtained approximation for the box-counting dimension of this figure is 1.57.

Remark 5.6. This result is less than the result obtained for the coastline of Great Britain, which implies that the borders of Luxembourg are "smoother", i.e. they have less edges than the coastline of Britain. This is in fact noticeable when looking at the pictures of both countries. However, this clearly isn't the "real" approximation of the dimension of Luxembourg, because this dimension is larger than the real dimension of Britain, which would contradict the observation that Luxembourg is "smoother".

All the images we used until now, only contained the outline of the countries, but we don't always have good pictures of a country that contains only the outline. Let us check what happens if we choose a picture containing more information, for example where all the cantons are included and the country itself is coloured.

Example 5.7. Let us choose for example the following picture of Luxembourg.

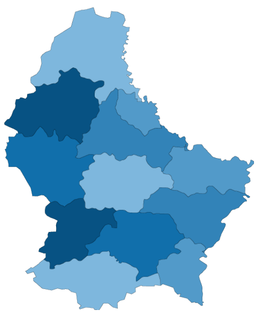


Figure 13: Luxembourg with a subdivision into cantons, [13].

For this picture of Luxembourg, we obtain a dimension of 1.60.

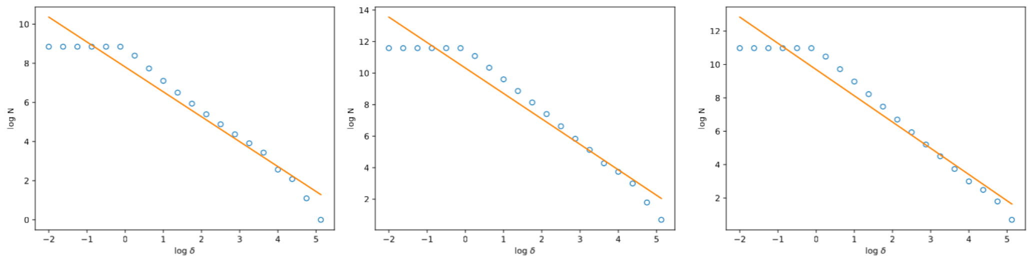
Remark 5.8. This dimension is larger than the dimension of Luxembourg in Figure 12. The reason for this difference is either because of the picture, that it has different borders or a different quality, or because the program added the borders of the cantons and maybe even the blue coloured parts to the parts of the image of which we want to calculate the dimension.

We see that this algorithm works but it still has some issues that we will discuss in the following subsection.

5.1.2 Problems

The algorithm we looked at in this section has several problems. One of these issues, which is also the most important one is that the obtained dimension is not precise. One of the reasons for this, may be that the quality of the pixels is not good enough. Another reason could be that the outlines in the image don't actually correspond to the real outlines of the figure in real life, for example, if the edges of a country are not represented correctly.

Another problem, that could also be one of the reasons why the resulting dimension is not precise, is that the first few values of the number of bins don't change. In order to visualise this observation, let us look at the graphic representation of the linear regression that the program gives us.



(a) (b) (c)

Figure 14: Linear regression of the circle (a), of Britain (b) and of Luxembourg (c). In the three images, the blue circles represent the points that have as coordinates the logarithms of the scales and the logarithm of the number of bins, and the orange line represents the linear approximation of these points.

Just as in the table obtained from the algorithm, we notice that, for small scales, the logarithm of the number of boxes seems to be constant. The reason for this could be that, the scales have become smaller than the size of a pixel, but the algorithm cannot consider particles that are smaller than one pixel. This would imply that it continues to consider the smallest scale possible and hence it counts the same number of bins containing the same data.

Another issue is that the included picture has to already contain nothing but the outline of the figure. This means that, for example in order to find the box dimension of the borders of Luxembourg, the image cannot have, for example, a subdivision into cantons or anything similar, one example that shows this issue is Example 5.7.

The next suggestion, which is not an issue, is that, in order to have a better visualisation of what is happening in the algorithm, it would be nice if the algorithm returned pictures where the subdivisions and the counted squares are shown.

In the next section, we will improve this algorithm, and find better approximations of the box dimension.

5.2 Box-counting algorithm that uses a plotting of the image

In this section, we will implement an algorithm that, hopefully gives a better approximation of the box dimension, than the previous one. In order to program it, we start by trying to plot the box-counting method, that is by drawing a grid over a given image and then colouring all the squares that are non-empty. The reason why we start by illustrating the box-counting method is that, if there is an issue, we find the error in the algorithm more easily, because we see what is wrong in the plotting. After that, we continue by counting the coloured squares and finding a linear regression of the logarithm of the number of squares compared to the logarithm of the size of the squares.
The algorithm we construct based on this idea is the following.

```python
import cv2 as cv
import numpy as np
from numpy import *
import pylab as pl
from PIL import Image
from PIL import ImageFilter
from PIL import ImageDraw
from PIL import ImageOps
import matplotlib.ticker as plticker
from matplotlib import pyplot as plt
import matplotlib.patches as mpatches


############################### Step 1: Include and edit the
    picture
Picture='name.png'

img = Image.open(Picture).convert('1')  #convert image to grayscale
img.save('img.png')

image=pl.imread('img.png')              #read image , make array
im=plt.imshow(img)                      #plot image
ax = plt.gca()                          #add axes
plt.show()

############################### Step 2: Creating a list with the
    scales
scales=[]
scales=[int(min(image.shape[1],image.shape[0])/l) for l in range
    (3,100,2)]
scales=list(dict.fromkeys(scales))
for i in range(len(scales)):
    if scales[i]==1 : scales.remove(scales[i])

```

```
################################ Step 3: Draw the grid, colour and count the non-empty squares
N=[]
for k in scales:                                    #Grid
    xticks = np.arange(0, image.shape[1] , k)
    yticks = np.arange(0, image.shape[0] , k)
    xt=ax.set_xticks(xticks)
    yt=ax.set_yticks(yticks)
    ax.grid(True, color='black', linestyle='-', linewidth=1)
    ax.patches=[]

    n=0                                              #colour and count squares
    for i in range(len(xticks)-1):
        for j in range(len(yticks)-1):
            l=xticks[i]
            m=yticks[j]
            for (x,y) in [(x,y) for x in range(xticks[i],xticks[i+1],1) for y in range(yticks[j],yticks[j+1],1)]:
                if image[y,x]==0:
                    s=ax.add_patch(mpatches.Rectangle((l,m), k, k, facecolor='red',edgecolor="none", linewidth=1))
                    n+=1
                    break
                else: continue
    plt.show()
    print('The number of squares trough which the image passes is:',n,'.\n')
    N.append(n)

table([(scales[i],N[i]) for i in [0..len(N)-1]], frame = True, header_row=['Scale','N'])

################################ Step 4: Find box-dimension
Polyfit = np.polyfit(np.log(scales),np.log(N),1) #given by log(N) = Polyfit[0]*log(scales)**1 + Polyfit[1]
print('The fitting polynomial of N is:', Polyfit[0],'x + ', Polyfit[1],'\n')
print('Hence, the box dimension is equal to' ,-Polyfit[0],'.\n')

################################ Step 5: Plot linear regression
plt.clf()
pl.plot(np.log(scales),np.log(N), 'o', mfc='none')
pl.plot(np.log(scales), np.polyval(Polyfit,np.log(scales)))
pl.xlabel('log $\\delta$')
pl.ylabel('log N')
plt.show()
```

Listing 4: Algorithm that plots the box-counting method.

This algorithm consists of five steps:

Step 1: **(Lines 14-24).** The aim of this step is to include and edit the input picture, so that later we can add grids and squares over it. In line 15, we simply include the picture of the figure of which we want to calculate the dimension and name it `Picture`. Next, we use the function `Image.convert(mode)`, that returns an image that is converted to the desired `mode`. We choose the mode `'1'`, which image is a black and white picture, that is saved to our files in line 18. Then, in line 20, the algorithm "reads" the image and creates an array named `image`, that contains all the information, such as the length, the height, and the colours (the values of the pixels), of the image. In the lines 21 to 23, we create a plot that contains the image and axes.

Step 2: **(Lines 25-31).** In the second step, we create a list containing all the scales, i.e. all the side-lengths of the squares that we will draw over the image. In order to create this list, we start by constructing an empty list named `scales`. The items we add to this list are given by the minimum of the length and the height of the image divided by some integer `l`, which takes all the values from `3` up to `100` with a step of `2`. For simplification, we want our scales to be natural numbers, but the results of the divisions that define the scales, are not necessarily integers. For this reason, we choose the integer that approximates this result best, by writing `int()`. In line 27, we add those integers to the list `scales`. However, these scales are approximations of the results of divisions, which implies that it is possible that some scales are repeated. Those repeated elements must be deleted, by writing `list(dict.fromkeys(scales))`. It is important to delete the repeated elements, because else we will consider some scales twice or even more often, which will falsify the box dimension obtained at the end. Also, the value 1 cannot be in the list because it is too small, it is impossible to create a grid with squares of side-length 1 pixel. That is why, in the lines 29 and 30, we create a `for`-loop, where the variable `i` goes over all the elements in the list `scales` and if one of the elements is equal to `1`, then this item is removed from the list, by writing `scales.remove(scales[i])`.

Step 3: **(Lines 32-58).** This step is the most important of all steps. Here, we draw the grids with squares of different sizes, that we colour and count if they contain parts of the figure. We start by creating an empty list, that will later contain the number of coloured boxes at each scale.

Next, we come to the main part of this step, that is the `for`-loop with variable `k`, that goes over all the items in the list `scales`. In this loop, we draw from line 35 to line 39, a grid consisting of squares of side-length `k`. In the lines 35 and 36 we create two arrays that contain the information for the positions of the lines of the grid. Then, in the lines 37 to 39, we set these values as the ticks, i.e. the positions of the shown values on the axes and draw the grid, whose lines pass trough all of these values, coloured black. In line 40, we set the `ax.patches` to be empty, in fact these patches are the squares that will be coloured, but we reset them to be empty, so that for every new value of `k` there are no patches from previous values.

In line 42, we set `n=0`, this will be the number of coloured squares. From line 43 to line 52, we enter a double `for`-loop with variables `i` that goes from 0 to the last tick on the x-axis and `j`, starting at 0 and ending at the last tick of the y-axis. In this double-loop, we set `l` to be the value of the $\texttt{i}^{th}$ tick on the x-axis and `m` will be the value of the $\texttt{j}^{th}$ tick on the y-axis.

In order to understand the next part, let us remark that by "the square on position `(l,m)` of the grid", is meant the square that goes from the $\texttt{i}^{th}$ to the $(\texttt{i+1})^{th}$ tick on the x-axis and from the $\texttt{j}^{th}$ to the $(\texttt{j+1})^{th}$ tick on the y-axis. Using this, it is easy to check that the `for`-loop starting in line 47, has as variable the couple `(x,y)` that goes over all the coordinates of the pixels in the square on position `(l,m)`.

In this loop, we consider the case `if image[y,x]==0`, that is, if the pixel with coordinates `(x,y)` is black. If this is the case, then, in line 49, we add a red coloured patch ("sticker") that is shaped like a rectangle of length and width `k` on position `(l,m)`. The rectangle that describes this patch is simply the square on position `(l,m)` of the grid. In addition, in this case, we increase `n` of 1, this means that for each square that is coloured, `n` increases by 1. In line 51, we `break` the loop, so that we only consider the first non-white pixel in the square because else, we would not count the number of coloured squares, but the number of coloured pixels, which is clearly larger.

If this case is not true, that is if the pixel at position `(x,y)` is white, then we continue with the loop.

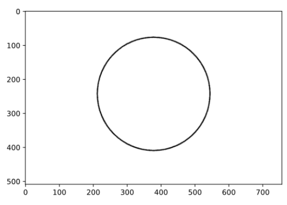
In the lines 53 to 55, we simply show the plot that is obtained after adding the patches, we print a sentence giving the number of squares that have been coloured and we add `n` to the list `N`. The last part of this step consists of creating a table containing the scales and the number of coloured boxes.

Steps 4-5: **(Lines 59-70).** The last steps of this algorithm are exactly the same as the last steps of the algorithm given in Listing 3.

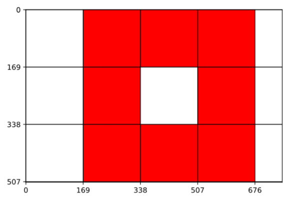
5.2.1 Outputs

For the same reasons as in Section 5.1, we start by testing the algorithm for a circle.

Example 5.9. Choosing the same picture as in Example 5.2, we will obtain an output with more than ten pictures. Including every picture here below would take a lot of space, therefore we only add some of the most interesting pictures.

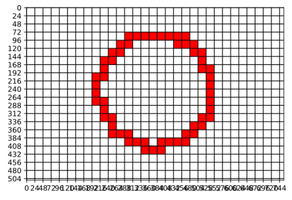


(a)



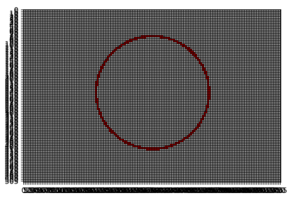
```
The number of squares trough which the image passes is: 8 .
```

(b)



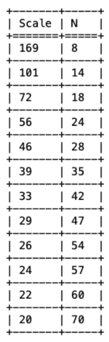
```
The number of squares trough which the image passes is: 57 .
```

(c)



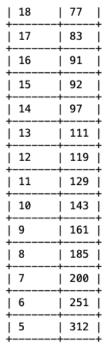
```
The number of squares trough which the image passes is: 312 .
```

(d)



```
+-------+-----+
| Scale | N   |
+=======+=====+
| 169   | 8   |
+-------+-----+
| 101   | 14  |
+-------+-----+
| 72    | 18  |
+-------+-----+
| 56    | 24  |
+-------+-----+
| 46    | 28  |
+-------+-----+
| 39    | 35  |
+-------+-----+
| 33    | 42  |
+-------+-----+
| 29    | 47  |
+-------+-----+
| 26    | 54  |
+-------+-----+
| 24    | 57  |
+-------+-----+
| 22    | 60  |
+-------+-----+
| 20    | 70  |
+-------+-----+
```

(e)



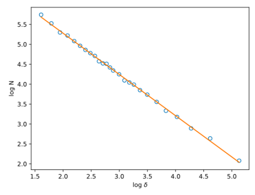
```
| 18    | 77  |
+-------+-----+
| 17    | 83  |
+-------+-----+
| 16    | 91  |
+-------+-----+
| 15    | 92  |
+-------+-----+
| 14    | 97  |
+-------+-----+
| 13    | 111 |
+-------+-----+
| 12    | 119 |
+-------+-----+
| 11    | 129 |
+-------+-----+
| 10    | 143 |
+-------+-----+
| 9     | 161 |
+-------+-----+
| 8     | 185 |
+-------+-----+
| 7     | 200 |
+-------+-----+
| 6     | 251 |
+-------+-----+
| 5     | 312 |
+-------+-----+
```

(f)

```
The fitting Polynomial of N is: -1.0349447983625306 x +  7.3485031838273684

Hence, the box dimension is equal to 1.0349447983625306 .

[<matplotlib.lines.Line2D object at 0x7fd9f578eb50>]
[<matplotlib.lines.Line2D object at 0x7fd9f578ee20>]
Text(0.5, 0, 'log $\\delta$')
Text(0, 0.5, 'log N')
```



(g)

Figure 15: Output of the algorithm for a circle. In (g), the blue dots are the points where the x-coordinates are the logarithms of the scales and the y-coordinates are the logarithms of the corresponding number of coloured squares, and the orange line represents the linear approximation of these points.

The output of the algorithm consists of several images with different grids. The squares of these grids that contain parts of the circle are coloured red and we see that on the last output image, the red squares almost look like the circle itself. After every image, comes a sentence describing how many squares are red. The algorithm also gives a table with the number of coloured squares `N` and the corresponding scales. Below the table follow two sentences, one giving the equation of the fitting polynomial and one giving the box dimension of the circle. Finally, there is a graphic representation illustrated in Figure 15g.

Remark 5.10. We observe that this algorithm gives a box dimension of approximately 1.03, which is very good. The error of the obtained dimension compared to the real dimension is only 0.03, which is almost optimal, compared to the error that we obtained in the previous algorithm.

It seems like this algorithm gives us the right approximation of the box dimension, so let us test it for a two more figures, before applying it to Luxembourg.

Example 5.11. The two figures for which we will test the algorithm are the coastline of Great Britain and the Sierpiński triangle. For the British coastline, we use the same image as in the example for the previous algorithm, namely Figure 11 and for the Sierpiński triangle, we use the following picture.

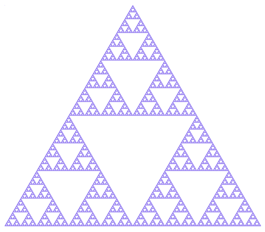
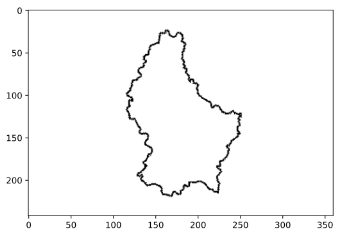


Figure 16: Image of the Sierpiński triangle, [14].

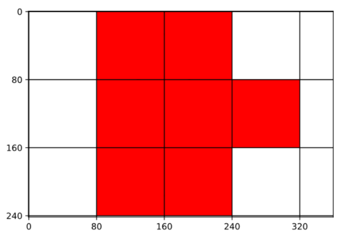
The algorithm returns a box dimension of 1.26 for the coastline of Britain and 1.58 for the Sierpiński triangle.

Remark 5.12. We know that in reality, Britain has a box dimension of 1.25 and from Example 4.6, we know that the dimension of the Sierpiński triangle is equal to $\frac{\log(3)}{\log(2)} \approx 1.58$. Hence, even for more complex, more edgy and less smooth figures, the algorithm returns dimensions that are very close to the real values.

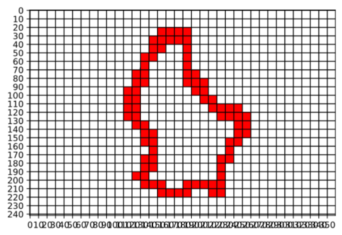
Example 5.13 (Applying the algorithm to Luxembourg)**.** Since this algorithm seems to be working really good, we can apply it to the borders of Luxembourg. If we run the algorithm with the same picture as in Example 5.5, that is Figure 12, we obtain the following output.



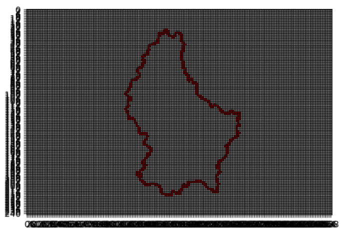
(a)



(b)



(c)

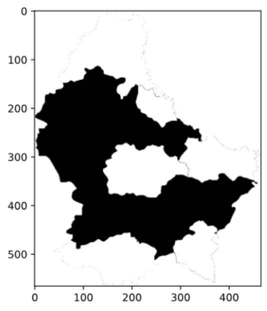


(d)

Figure 17: Some of the output images of the algorithm for the borders of Luxembourg.

The first observation we make is that the figure created by the red squares approaches the shape of Luxembourg very accurately. This implies that the box-counting dimension obtained for the borders of Luxembourg, which is 1.05, is a good approximation. We also notice that this value is close to 1, which means that the borders of Luxembourg don't have a lot of deep edges.

Example 5.14. As in Example 5.7, let us check if the algorithm still returns a good approximation if the image contains more information, such as names or subdivisions. We chose the same image, that is Figure 13, and get the following two images in the output.



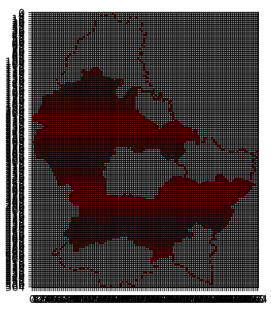


Figure 18: Two of the output images of the algorithm for Luxembourg with subdivision into Cantons.

We see that when converting the input image to a black and white image, the fields that are coloured in a darker blue become black. This implies that, when colouring the squares through which the figure passes, the algorithm also considers these black fields. Since the algorithm calculates the linear regression with a larger number of squares compared to the number of squares obtained by only colouring the borders, the obtained box dimension is also larger, it is 1.72.

After having described the algorithm and having looked at the outputs of it, we can look at the problems that it still has.

5.2.2 Problems

The algorithm gives pretty good approximations of the box dimension, which implies that there are not a lot of problems. However, the program is not perfect. The first problem that we notice is that in an image that has small borders, not every square that contains the figure is coloured. One example where this happens is for Figure 11 of Britain.

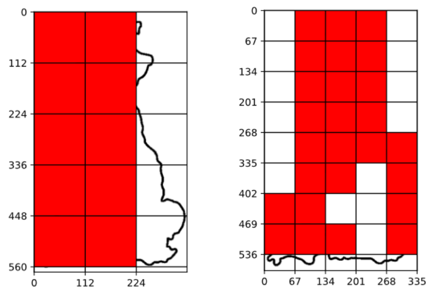


Figure 19: Two of the images of the output for Figure 11.

In the two pictures above, we see that the last row and the last column of squares have not been coloured, which means that they have not been considered. Since this falsifies the final result, this problem needs to be solved. We notice that this mainly happens for larger squares, since for smaller squares, there is enough space in the border for the squares to be white. Thus, to solve this issue, we start with a smaller scale, which doesn't fake the end result, since for the linear regression we do not need every scale.

Another problem that occurs is that the input picture can only contain the figure and no other annotation, dark colouring or subdivision. This problem has already occurred in the algorithm in Section 5.1. Let us see what other problems they have in common, by comparing them in the next section.

5.3 Comparing the two algorithms

First, we compare the codes of the algorithms, where the main difference is that one algorithm works with an array that describes the image and evaluates the information with a histogram and the other one plots the image and modifies the plotting to find the squares containing parts of the figure. To compare the outputs of the two algorithms, let us consider the following table.

	Circle	Great Britain coastline	Luxembourg borders	Luxembourg with cantons
Algorithm 1	1.27	1.61	1.57	1.60
Algorithm 2	1.03	1.26	1.05	1.72
Exact value	1	1.25	?	?

Table 3: The obtained box dimensions in the two algorithms.

We see immediately that the algorithm of Section 5.2 gives better approximations of the box dimension than the algorithm of Section 5.1. Another amelioration of the output, that is not visible in this table, is that the second algorithm looks nicer because it returns pictures where it is visible how the figure is approached.

6 Conclusion

The goal of this thesis was to understand the notion of box-counting dimension and then find the box dimension of the borders of Luxembourg. To do this, we started by introducing the Coastline paradox that explains why we use fractal dimension, and we also introduced the Hausdorff dimension, which is one of the fractal dimensions. After having introduced these notions, we defined the box-counting dimension, we proved some properties and we looked at some examples by calculating the box dimension of some figures, such as the Sierpiński triangle or the middle third Cantor set.
The notion of box dimension allowed us to understand the box-counting method, which is a technique that can be used to get the box dimension of given figures. We used this method to program two algorithms that allowed us to calculate the box dimension of the borders of Luxembourg. The first algorithm didn't give precise enough solutions, that is why we know that the dimension for the borders of Luxembourg obtained from this algorithm is not correct. However the second algorithm gave good approximations, and we obtained a box-counting dimension of approximately 1.05 for the borders of Luxembourg.

Even if the second algorithm seems to have completed the task, this project is not entirely over yet. One could still improve this algorithm by trying to make it work for images containing more information than just the figure. This can be done by editing the input image more, so that all these details disappear after the conversion.

Another idea to continue this project is to apply Proposition 4.4, which states that instead of boxes, we can take whatever object we want. One could try to create an algorithm that approaches the image with circles, triangles, hearts or even stars. Every shape can be used for this experimentation.

Also, the box-counting dimension is not only defined for objects who's dimension is between one and two, i.e for figures that can be represented in a picture. An idea would be to try to find an algorithm that allows us to calculate the box dimension of three- or more-dimensional objects.

We can see that there is still a lot to explore and a lot of experimentations that one can do with this box-counting dimension.

References

[1] Kenneth Falconer. *Fractal Geometry - Mathematical Foundations and Applications (Third edition)*. University of St Andrews, UK 2014.

[2] Coastline paradox. *Wikipedia*, April 2021. Page Version ID: 1020705980.

[3] Jono Hey. *Sketchplanations*.
https://sketchplanations.com/the-coastline-paradox.

[4] How Long Is the Coast of Britain? Statistical Self-Similarity and Fractional Dimension. *Wikipedia*, November 2020. Page Version ID: 988885247.

[5] Sinead Sukerta. *Youtube*. https://youtu.be/zfcVVweWFQI.

[6] Lipschitz continuity. *Wikipedia*, February 2021. Page Version ID: 1009048104.

[7] Similarity (geometry). *Wikipedia*, April 2021. Page Version ID: 1017264704.

[8] Isometry. *Wikipedia*, April 2021. Page Version ID: 1020387069.

[9] Affine transformation. *Wikipedia*, March 2021. Page Version ID: 1011191615.

[10] Francesco Turci. *francescoturci.net*, March 2016.
https://francescoturci.net/2016/03/31/box-counting-in-numpy/.

[11] John Hoggard. *How long is the Coast of Great Britain?*, May 2021.
http://www.aiecon.org/staff/shc/course/annga/RR/main/How%20Long%20is%20the%20Coast%20of%20Great%20Britain.htm.

[12] Dominik. *Shutterstock.com*, Image number: 1163284309.
https://www.shutterstock.com/fr/search/luxembourg+outline.

[13] *Webvertormaps.com*, 2021.
https://webvectormaps.com/blank-map-of-luxembourg-by-cantons/.

[14] Sierpiński triangle. *Wikipedia*, May 2021. Page Version ID: 1021110752.