Zahra Iqra Hussain 170993054

GENERATING FRACTALS

Contents

Abstract

The purpose of this project is to explore concepts within fractal geometry, including the box counting dimension and self-similarity. This will provide us with a smooth link to iterated function systems, which will focus on how fractals can be generated.

1. Introduction

The term ‘fractal’ was created by Benoit Mandelbrot in the 1970s, and was used to describe a ‘jagged’ or ‘broken’ set.

Natural shapes such as plants and coastlines can be interpreted as fractals. They can also be represented as curves whose fractal dimensions are greater than one. They are fairly irregular, so we must use a range of methods to accurately evaluate such lengths.

Throughout this essay, we will consider aspects within ‘Fractal Geometry’. We will begin by gaining a fundamental understanding of dimension and its importance. Thereafter, we will bring our attention to the box counting method. By defining this concept and finding its general equation to compute dimensions, we will then move on to look at some of the properties it satisfies. This will motivate us to apply it to some fractal examples. Studying self-similarity in chapter 3 will allow us to do this.

We will compute the lengths and dimensions of several fractals, including the Koch curve, Cantor set and Sierpinski triangle. Since coastlines will be heavily discussed in chapter 2, it will also be useful to study their self-similar nature.

Finally, using these examples, we will study the mathematical framework behind iterated function systems. By creating simple recursive equations, we will show how these fractal structures can be constructed. This will be covered in chapter 4.

2. Box Counting Dimension

2.1. Dimension

We can define dimension as a topological measure of the size of its covering properties[1]. It is able to determine the number of points needed to cover an object. Thus, it is vital to geometry; more specifically fractal geometry.

Dimension is seen as an essential tool in characterizing fractal structures. It measures these objects by looking at how complicated their self-similar figure is. By quantifying the complexity of a fractal pattern, we can see how dimension changes depending on the scale at which the fractal is measured. In chapter 3 we will see how the dimension of fractals becomes more intricate, as rectangles with smaller side lengths are used to measure them.

Alongside this, dimension is significant to transformations. Regardless of any transformation(s) made to an object, their dimension never changes. This includes rotation, reflection or translation. We will show evidence of this in chapter 4, where we use rotation and reflection matrices to create an iterated function system for the Koch curve.

Fractals tend to have non integer dimensions, as we will see in subsection 2.4. However, we can extend this to a range of sets. For example, a smooth curve is 1-dimensional, a plane is 2- dimensional and so on. Within general mathematics we often work with the 3-dimensional space – the Euclidean space. This will be considered when introducing the box counting dimension in subsection 2.2.

The box counting method is used to calculate the dimensions of fractals. We will begin by understanding its framework.

2.2. Mathematical concept of the box counting dimension

The box counting dimension is a concept of dimensions for fractals. It involves looking at the fractal structure, and the patterns it is made up of. By breaking it up into smaller fragments and measuring it by the number of boxes with a specific side length, this method is able to compute limits to fractal dimensions. We can also use it for simple geometric objects.

We can introduce this notion by looking at a simple example:

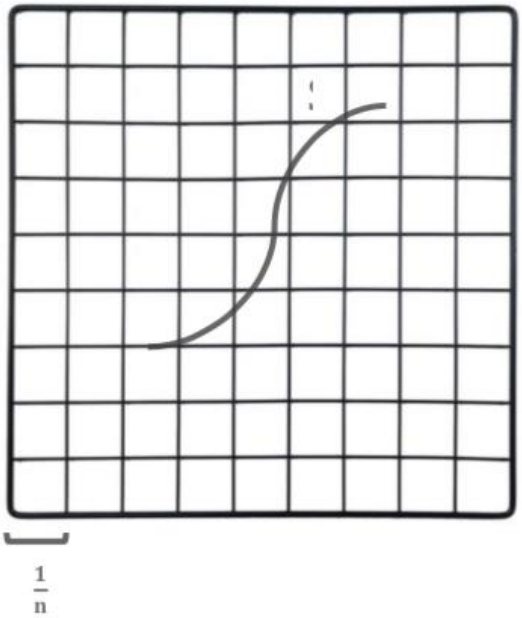

Fig 2.2.1: Curve S in a unit square[2]

1) We begin by assigning a dimension to a set S where $S \subset \mathbb{R}^n$. For simplicity, we consider the case where S is a curve in $\mathbb{R}^1$.
2) Assuming $S \subset I \times I$ (a unit square), we divide it into smaller, identical squares. The side length (ε) of each of these squares is equal to $\frac{1}{n}$.
3) Let $N_n = N_\varepsilon$ be the number of squares with side length ε that intersect S. Then, for the curve S (shown in fig 2.2.1) we can see $N_n \sim n$ as $n \longrightarrow \infty$.

Let us look at another example.

As mentioned in subsection 2.1, within mathematics we often work with the Euclidean space. Thus, it would be useful to see how the box counting method would work for different dimensions in this space.

By referring to the side length of a square as 'ε', we can illustrate dimensions 0, 1, 2 and 3 through Euclidean objects. We find the size of these objects by counting the number of squares (N_ε) used to cover them. This is demonstrated in fig 2.2.2:

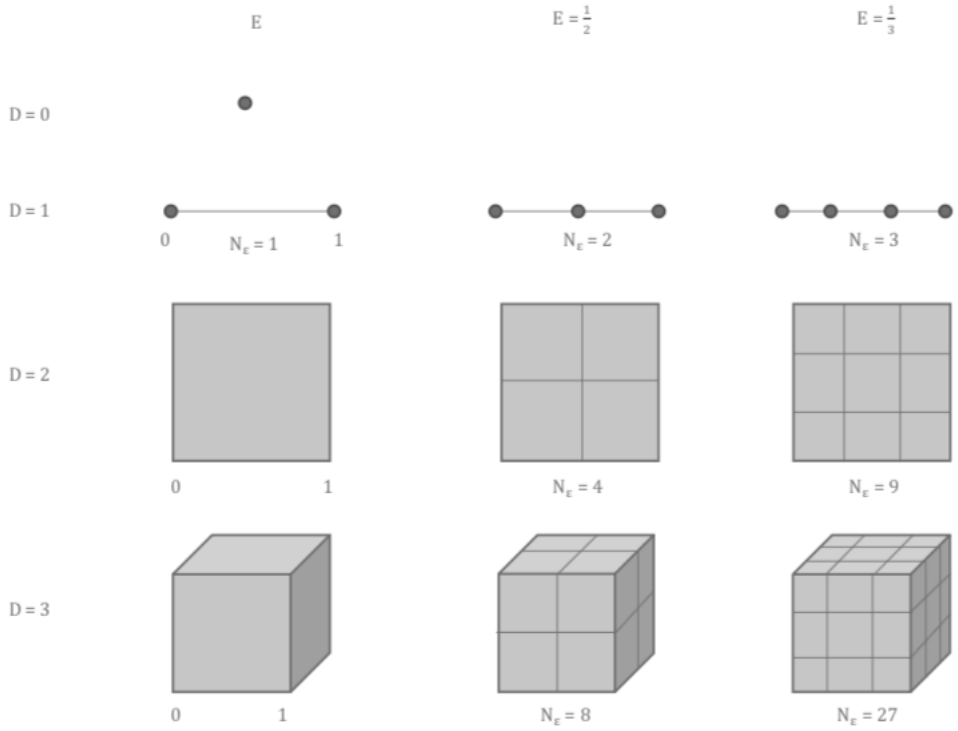

Fig 2.2.2: Box counting dimension using Euclidean objects[3]

Looking at fig 2.2.2, we can see that the number of squares of side length $1/2$ needed to cover a two dimensional object is 4. In contrast, the number of squares of side length $1/3$ needed to cover the same two dimensional object is 9. The same trend follows for all other objects of different dimensions. For example, measuring the three dimensional cube with squares of side length $1/2$ requires 8 squares, whilst measuring the same cube with squares of side length $1/3$ requires 27 squares. So as we use squares with smaller side lengths, more are needed to cover the different geometric shapes.

From this, we can consider a simple algorithm: the number of squares of side length $1/\varepsilon$ needed to cover an object with dimension D is equal to ε^{D}. This allows us to relate dimension with length.

Having partially studied the nature of box counting through our examples above, we can now come to terms with a more formal definition:

Let S be a bounded subset of $\mathbb{R}^n$ and set $N(S,\varepsilon)$ to be the number of rectangles of side length ε needed to cover S. Then the **box-counting dimension** of S is defined to be

$$\mathrm{Dim}(S) = \lim_{\varepsilon \to \infty} \frac{\ln\left(N(S,\varepsilon)\right)}{\ln\left(\frac{1}{\varepsilon}\right)} \qquad (2.2.1)$$

This equation is only defined if the limit exists. A more complicated calculation, the Hausdorff- Besicovitch dimension, always exists for a bounded subset of $\mathbb{R}^n$ and usually agrees with the box counting dimension.

We should also note here that fractal dimensions tend to have non integer values. This will be explored further in subsection 2.4 and chapter 3.

We can use the box counting method to compute several fractal dimensions. But before that, it would be useful to look at its properties to understand its background a bit more.

2.3. Properties of the box counting dimension

The following list[4] covers some basic properties satisfied by this concept:

1. The box counting dimension is monotonic. This means if $E \subset F \subset \mathbb{R}^d$ then $\dim_B(\varepsilon) \leq \dim_B(F)$.
2. If F is an open subset of $\mathbb{R}^d$, then $\dim_B(F) = d$.
3. If F is non-empty and finite, then $\dim_B(F) = 0$. This is because if F contains m distinct points, then $N_\delta(F) = m$ for all small δ.

By considering the first property, we can reach a simplified conclusion. That is, if E is a subset of F, its dimension will be less than that of F. This is because E is contained within F.

Additionally, we can provide clarification to the second property; the dimension of F is equal to the dimension of the set it is contained in. This is only if it is an open subset within this set. For example if F is an open subset of $\mathbb{R}^2$, its dimension will be equal to two. This is because it is open, so it can produce a range of points. However, these points can only be contained in $\mathbb{R}^2$.

Finally, the third property explains that if F consists of m distinct points, its dimension would be 0. This is because points are 0-dimensional. The number of squares of side length δ required to cover these points would be equal to m (the number of points) as they are distinct from each other.

Now that we have understood the box counting dimension, we can put it into practice. We can begin by considering coastlines.

Coastlines can be considered as fractal curves, since they tend to hold their same pattern of irregularity, regardless of how much they are magnified (illustrated in fig 3.4.1). Thus, their dimension can be computed using the box counting method.

2.4. Example: Measuring the coastline

When describing the size of irregular shapes, such as the coastline, the box counting method proves useful.

For example, we can start by setting dividers to a specific opening ε. These can be labelled as the yardstick lengths. We can then walk these dividers along the coastline with each step starting where the previous left off, giving us the number of walks of side length ε needed to cover the coastline. This can be illustrated as follows:

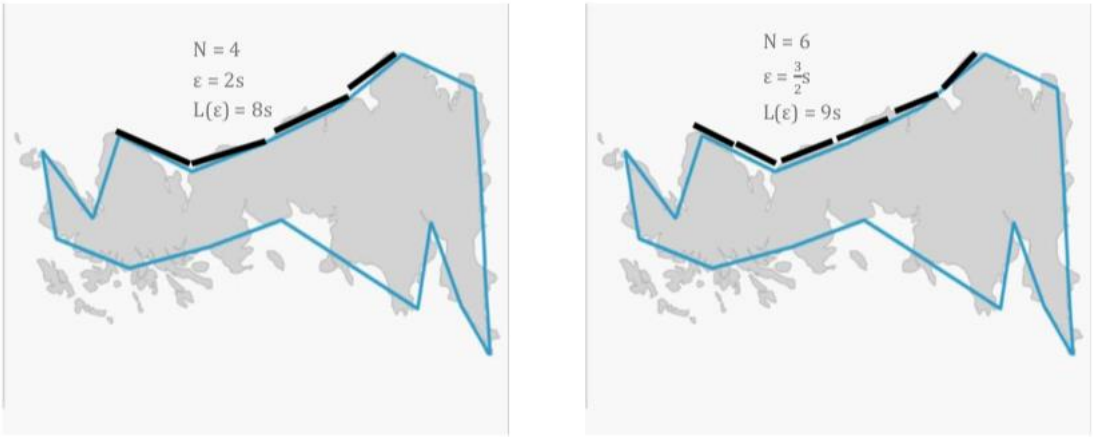

Fig 2.4.1: Measuring a section of Britain's coastline[5]

The approximate length of the coastline L(ε) can be calculated by multiplying the number of steps N by ε. One would naturally expect that as the opening of the dividers' become smaller, L(ε) would settle to a clear true length. However, looking at fig 2.4.1 we can see that this does not occur. By annotating the figure, we have realised that the length of the coastline tends to increase. For example, using a yardstick length that is 2s would produce an overall coastline length of 8s. On the other hand, using a smaller yardstick length that is $\frac{3}{2}$s would produce a coastline length of 9s. So as we are using smaller yardstick lengths, the length of the coastline is increasing. We are not reaching a finite value. Why is this?

If we look at a bay on a map with a scale 1/100,000 and re-examine it on a map at 1/10,000, a number of sub bays will appear[6]. If we then take this map and re-examine it on a 1/1,000 scale map more will appear, and the process continues. This means that as we introduce finer details to the coastline, its overall length will increase. Fig 2.4.2 evidently shows this. We can also back up our assumption by the Richardson effect.

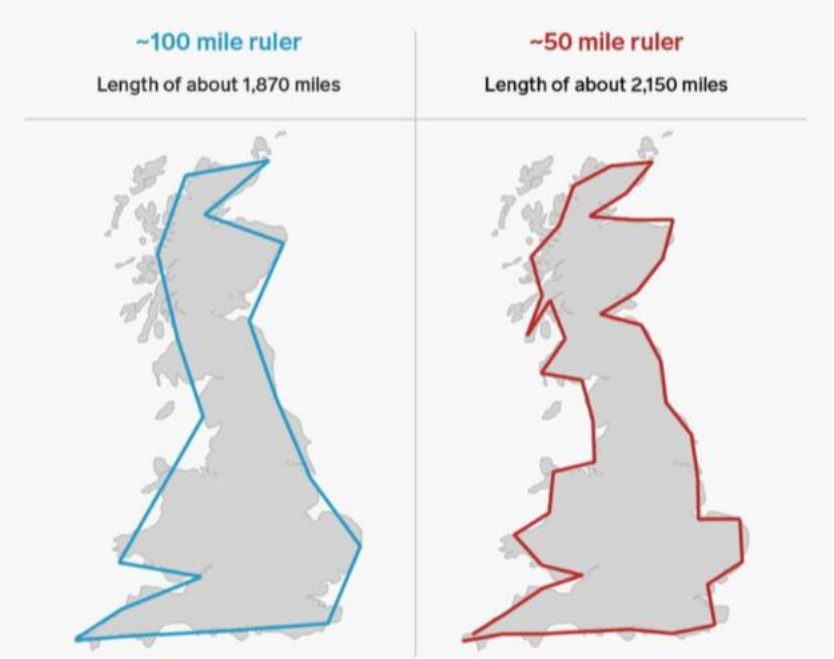

Fig 2.4.2: The coastline of Britain on different scales[7]

2.5. The Richardson Effect

Through various experimental measurements, Richardson demonstrated how the length of coastlines increased without limit as their yardstick length was made smaller. This is now known as the 'Richardson Effect'.

Using two constants, λ and D, Richardson deduced that approximately $F\varepsilon^{-D}$ intervals of length ε are required to add up so that our length $L(\varepsilon) \sim F\varepsilon^{1-D}$.[8]

Richardson's research was quoted by Benoit Mandelbrot through his paper 'How Long is the Coast of Britain?' in 1967. He came to conclude that coastlines could be seen as approximate fractal curves. The slope of each line can be used as an estimate of 1-D, where D is the fractal dimension. Thus, coastline dimensions obtained a value between one and two (not an integer).

The following is a diagram found among Richardson's papers after he died, and can be analysed to support our conclusion.

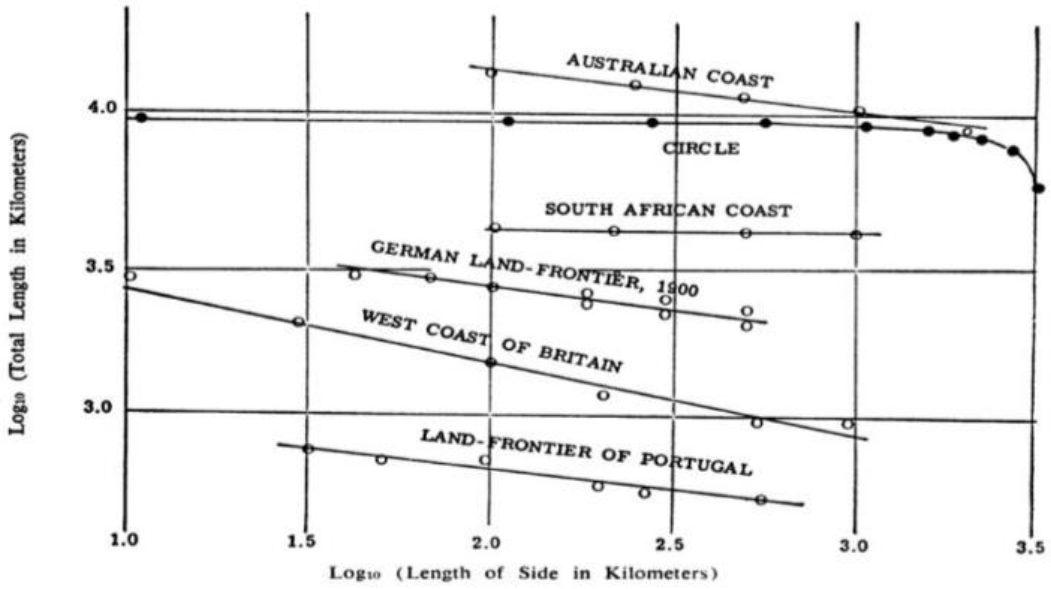

Fig 2.5.1: Richardson's empirical data regarding the rate of increase of coastline lengths[9]

Fig 2.5.1 accurately depicts the difference in steadiness of measurements taken on a circle in comparison to coastlines. The length of the circle seems to stabilize at a specific value, whilst the coastline lengths do not stabilize at all.

For the circle, we can assume that using a divider less than three kilometres would be accurate in providing a well determined value of the geometrical object. The South African coastline displays a near to zero slope, suggesting a somewhat smooth coastline. However the frontier of Portugal illustrates a negative slope as the unit length used is increased. Through different encyclopedia searches made by Richardson, the lengths of such frontiers claimed by regions within Portugal showed major differences (987 vs 1214km[10]). These could be explained by the various unit lengths used.

We can interpret fig 2.5.1 in a specific regard, to create a simple theory; as the slope on the plot becomes steeper, the length of the coastline increases more quickly. Therefore, their fractal dimension also increases.

2.6. Summary

Having studied the box counting dimension and its underlying concept, we saw that it could be an essential tool in computing the dimension of geometric objects. Considering the Euclidean space was also useful in introducing dimension on a broader scale.

Furthermore, from examining coastlines, we were able to realise that the box counting method could be used effectively in fractal structures. This example was helpful in providing us with considerable evidence to conclude that fractal dimensions do not need to be defined as integers. Instead they can be constituted in the realm of real numbers.

Researching into the Richardson effect supported this, deducing that coastline dimensions exist between 1 and 2.

We also noticed an interesting linear relationship between estimated length and side length when looking at the log-plot in fig 2.5.1. This can be generalized to fractal dimensions. For example, as smaller rectangles are used (with smaller side lengths), the dimensions of these objects get bigger.

In summary, the box counting method can be classed as one major property of fractals; another one being self-similarity. We will now study this through a few examples, whilst also computing their dimensions.

3. Self-similarity

Most fractals have some sort of self- similarity. This is because they are constructed by a repetitive rule. So, when rescaling them, we can see that they are made up of iterations of the same shape. This makes it easy for us to compute their dimension.

Throughout this chapter we will cover four different fractals to study self-similarity, one of which will be the coastline. We will also use the box counting method to calculate their dimensions.

3.1. Sierpinski's triangle

We can construct this geometrical object by taking a triangle with equal side length of one ($\varepsilon = 1$). We then divide this into four equal triangles and remove the triangle in the center to create a hole. This leaves us with three small triangles. We repeat this process for each of the three triangles as follows:

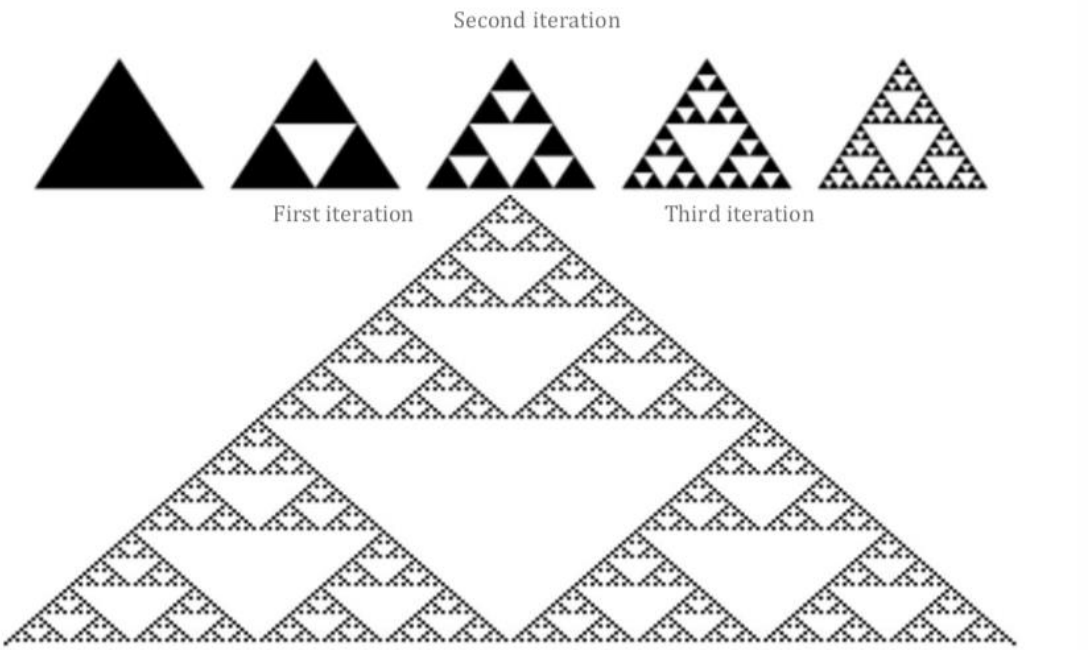

Fig 3.1.1: Sierpinski's triangle[11]

As shown in fig 3.1.1, the number of holes is increasing as we continue to iterate. This means the overall area of our triangle is becoming smaller and smaller. We can create a table to present the number of triangles removed in each iteration. We can also consider the total area of the removed triangles.

Number of triangles	1	3	3^2	3^{n-1}
Area of each removed triangle	$\frac{1}{4}$	$\frac{1}{4^2}$	$\frac{1}{4^3}$	$\frac{1}{4^n}$
Total area of removed triangles	$\frac{1}{4}$	$\frac{3}{4^2} = \frac{3}{16}$	$\frac{3^2}{4^3} = \frac{9}{64}$	$\frac{3^{n-1}}{4^n}$

We can calculate the area of the Sierpinski triangle by subtracting the area of each removed triangle from the area of the triangle we started with (equal to one):

$$1 - \frac{1}{4} - \frac{3}{4^2} - \frac{3^2}{4^3} - \ldots = 1 - \frac{1}{4}\left(1 + \frac{3}{4} + \frac{3^2}{4^2} + \cdots\right)$$

$$= 1 - \frac{1}{4}\frac{1}{1-\frac{3}{4}} = 1 - \frac{1}{4} \cdot \frac{4}{1} = 0$$

As we create infinitely many iterations, our area eventually becomes zero. Although this is evident from our illustration, it is useful to show our methods so we can establish the differences in our computations for area and dimension.

Dimension is different to area. The area is a measure of the extent of a surface – it is measured using square units. Dimension on the other hand, is a given characteristic of an object. It is a construct where objects can be distinguished. Length, area and volume are usually associated with 1,2, and 3 dimensions. Since fractals have non integer dimensions, it is not practical for us to link them with such structures.

We can now calculate the dimension of the Sierpinski triangle using the box counting method.

Firstly, we find N(S,ε) – the number of rectangles of side length ε needed to cover each triangle. The value of ε gets smaller as we split it up into smaller triangles. We do this for each iteration:

Step 0 : $N(S,1) = 1$

Step 1: $N\left(S,\frac{1}{2}\right) = 3$

Step 2: $N\left(S,\frac{1}{4}\right) = 3^2$

:

Step n: $N\left(S,\frac{1}{2^n}\right) = 3^n$

Hence using Equation (2.2.1),

$$\text{Dim}(S) = \lim_{n\to\infty} \frac{\ln N\left(S,\frac{1}{2^n}\right)}{\ln\left(\frac{1}{\frac{1}{2^n}}\right)} = \lim_{n\to\infty} \frac{\ln 3^n}{\ln 2^n} = \frac{\ln 3}{\ln 2}$$

$$= \log_2 3 \sim 1.58496\ldots$$

The dimension of the Sierpinski triangle is ~ 1.58. It is between 1-dimensional and 2-dimensional objects. So it is between a line and a plane, but a little closer to a plane. It is constructed by a repetition of the same triangle. Thus, it satisfies the property of being self- similar.

3.2. Koch Curve

Like the Sierpinski triangle, we construct this curve by starting with a line of interval length 1. We replace the middle third of the line by two lines of the same length, equal to $\frac{1}{3}$, and join them together at their center. This creates a 'ragged' like curve, consisting of four lines of equal length $\frac{1}{3}$ (fig 3.2.1), and an overall curve length equal to $\frac{4}{3}$.

We then repeat this process with each of the four lines, replacing the middle thirds of each line into two and joining them at their center. As we continue to iterate the curve, the length gradually increases. Thus, after infinitely many iterations, we obtain a geometric object with infinite length. This can be illustrated as follows:

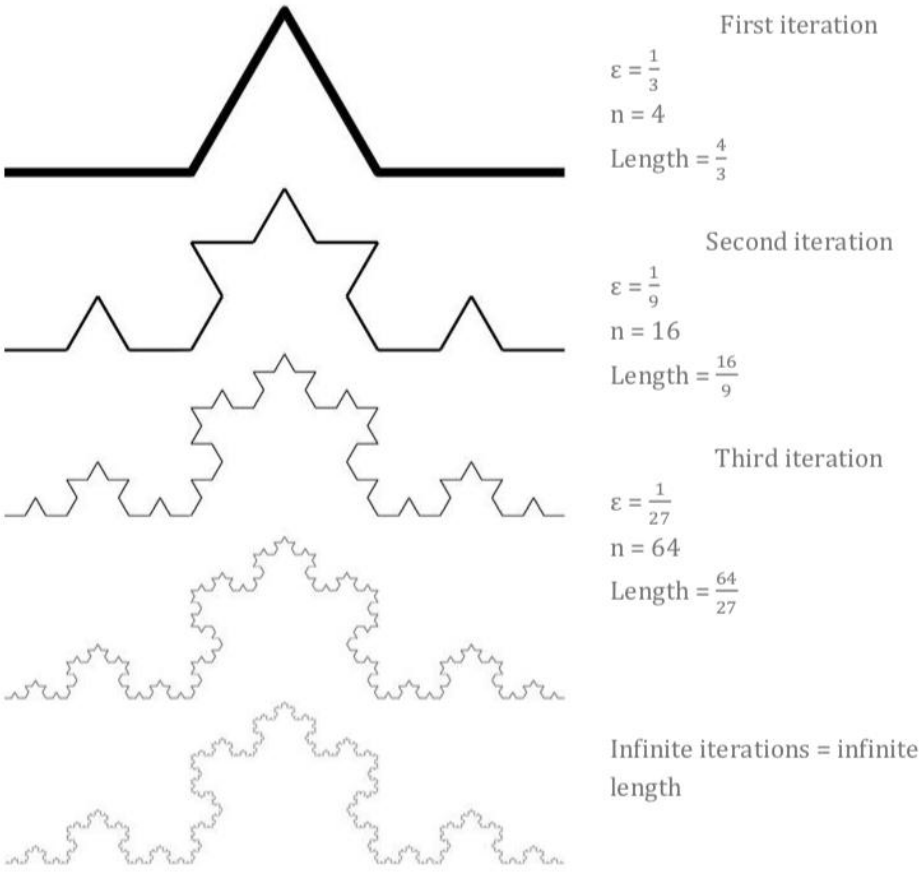

Fig 3.2.1: Koch Curve (all computations have been produced by me)[12]

We can measure the Koch curve inductively. At n iterations, we can gather that our length is equal to $\left(\frac{4}{3}\right)^n$. So as n tends to infinity, the length of our curve = $\lim_{n\to\infty}\left(\frac{4}{3}\right)^n = \infty$.

Using the box counting dimension, we can calculate the dimension of the shape:
We find N(S,ε) for the first few steps:
Step 0: N(S,1) = 1
Step 1: $N\left(S,\frac{1}{3}\right) = 4$
Step 2: $N\left(S,\frac{1}{9}\right) = 4^2$
:
Step n: $N\left(S,\frac{1}{3^n}\right) = 4^n$

Using Equation (2.2.1),

$$\text{Dim}(S) = \lim_{n\to\infty}\frac{\ln N\left(S,\frac{1}{3^n}\right)}{\ln\left(\frac{1}{\frac{1}{3^n}}\right)} = \lim_{n\to\infty}\frac{\ln 4^n}{\ln 3^n} = \frac{\ln 4}{\ln 3}$$

$$= \log_3 4 \sim 1.26186\ldots$$

The dimension of the Koch curve is 1.26 - slightly smaller than Sierpinski's triangle but still between 1-dimensional and 2-dimensional objects. Since each part is made up of the same structure, the curve is strictly self-similar.

3.3. Cantor Set

To construct this shape, we start with a unit length polygon. We then divide it into three subsections, and like the Sierpinski triangle, remove the middle subsection. This leaves us with two polygons of equal size. We then repeat this process with the two smaller polygons, removing their middle subsection after splitting into three. We continue to iterate, obtaining a structure as follows:

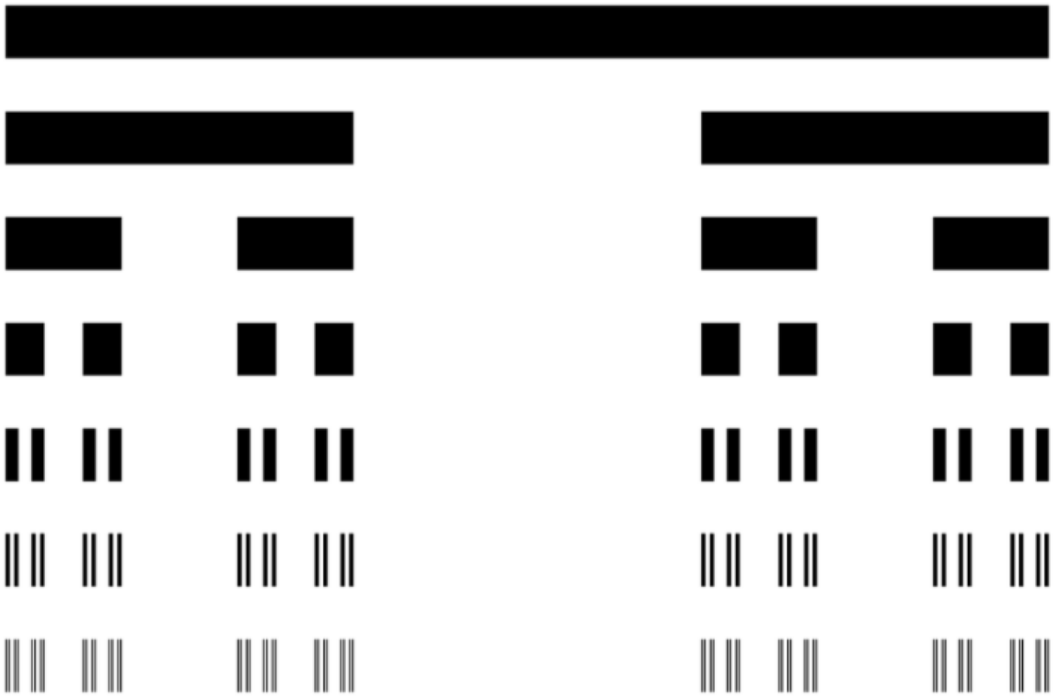

Fig 3.1.1: Cantor Set[13]

Using the box counting method, we can calculate the dimension of the Cantor set. We can produce our results for each step in a table:

Step	Length of intervals	Number of intervals
0	$\varepsilon = 1$	$N(C,1) = 1$
1	$\varepsilon = \frac{1}{3}$	$N\left(C,\frac{1}{3}\right) = 2$
2	$\varepsilon = \frac{1}{3} \cdot \frac{1}{3} = \frac{1}{3^2}$	$N\left(C,\frac{1}{3^2}\right) = 4 = 2^2$
3 :	$\varepsilon = \frac{1}{3} \cdot \frac{1}{3^2} = \frac{1}{3^3}$	$N\left(C,\frac{1}{3^3}\right) = 8 = 2^3$
N	$\varepsilon = \frac{1}{3^n}$	$N\left(C,\frac{1}{3^n}\right) = 2^n$

$$\text{Dim}(C) = \lim_{n\to\infty} \frac{\ln N\left(C,\frac{1}{3^n}\right)}{\ln\left(\frac{1}{\frac{1}{3^n}}\right)} = \lim_{n\to\infty} \frac{\ln 2^n}{\ln 3^n} = \frac{\ln 2}{\ln 3}$$

$$= \log_3 2 \sim 0.63093\ldots$$

The dimension of the Cantor set is 0.63 – between 0 and 1. The 0 dimension is a point, whilst 1 dimension is a line, thus the Cantor Set is slightly more closer to the dimension of a line. Creating infinitely many iterations shows us how self-similar this shape is.

3.4. Coastlines

Self-similarity can be viewed from a real-life perspective. For example, from looking at the coast of Britain on a range of scales, we can see that it is made up of approximate or exact repetitions of a specific structure.

Fig 3.4.1: Satellite map of a fragment of Britain's coastline on different scales[14]

These figures show a section of Britain's coastline taken at different scales. Comparing all three images, we can see that there is a comparable structure across the coast. We can assume that parts of the coastline are made up of the same, if not similar, fragments. Therefore many coastlines satisfy the self-similarity property.

3.5. Summary

Looking at the self-similar nature of fractals in a range of examples has allowed us to understand their structures further. For example, from researching Sierpinski's triangle, we saw how it was constructed. This allowed us to conclude, with evidence, that it is made up of smaller triangles of the same structure.

Similarly, we saw that the Koch curve contained multiple fragments which are similar and infinitely repeating. The Cantor set also illustrated subsets, which are similar in their iterations.

By considering the coastline as a fractal curve, we also looked into its geometry. Taking a section of the coast of Britain and studying it on a range of scales allowed us to realise that it too is self-similar.

Finally, using the box counting method to compute the fractal dimensions of these objects allowed us to see that the resulting dimensions of all our examples were not integers, but instead real numbers. We could relate this as a 'property' in fractal geometries; though we would have to prove this.

Since we can define fractals as objects that are self-similar, it would be useful to discuss how we can construct them. Using iterated function systems allows us to produce mappings that create these repetitive structures.

4. Iterated Function Systems

We can generate fractals using iterated function systems. This can be done by transforming each repeated fragment found in the fractal structure, into a function. Such functions are able to make shapes smaller and bring points closer together. These are known as contraction mappings.

4.1. Introducing iterated function systems

We can begin by understanding basic concepts within this topic:[15]

i) An iterated function system on $\mathbb{R}^d$ is a finite set $\{f_1, \dots f_n\}$ of contraction mappings. It can be written as follows: $f_i: \mathbb{R}^d \to \mathbb{R}^d$ where i=1…n

ii) The contraction mappings we produce (as we iterate) obtain a smaller distance than the overall distance of the function. This can be written in a more mathematical way:
$f:\mathbb{R}^d \to \mathbb{R}^d$ is a contraction if there is some $0 \leq k < 1$ such that
$dist(f(x), f(y)) \leq k \cdot dist(x,y)$ for all $x,y \in \mathbb{R}^d$

iii) Given an iterated function system $\{f_1, \dots f_n\}$ and $f_i: \mathbb{R}^d \to \mathbb{R}^d$, a contraction, the **limit set** is the set S^* satisfying $f_1(S^*) \cup f_2(S^*) \cup \dots \cup f_n(S^*) = S^*$.

iv) If k is any closed and bounded subset of $\mathbb{R}^d$, by applying F to k (where $F(k) = f_1(k) \cup f_2(k) \cup \dots \cup f_n(k)$) we have $f^n(k) \to S^*$ as $n \to \infty$.

How are iterated function systems related to fractals?
In essence, iterated function systems are contraction mappings which define closed invariant sets. Invariant sets are sets which remain unchanged, regardless of any modification made in the conditions that it is measured. These are often known as fractals.

Using this method allows us to construct fractals and adjust them without changing their structure. This is how their self-similar nature is obtained. The coefficients related to these systems enter the fractal dimension depending on the number of iterations we take.

4.2. Constructing fractals using iterated function systems

Iterated function systems may be built from non-linear functions such as Mobius transformations, but in our case we will use linear.

We can begin by considering the contraction mappings $F_m: \mathbb{R}^2 \to \mathbb{R}^2$.
By applying these to a vector S, we get a transformed vector S'.

If we look at an (x, y)-plane, we can consider S as cartesian coordinates of points in the plane. Then, we can write a general equation for these transformations:

$$S' = \begin{bmatrix} x' \\ y' \end{bmatrix} = c \begin{bmatrix} x \\ y \end{bmatrix} + c \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} cx + ca \\ cy + cb \end{bmatrix} = F_m(S) \quad (4.2.1)$$

Where a, b, c are constants. We can say c is the contraction factor, and $\begin{bmatrix} a \\ b \end{bmatrix}$ is a vector. We can use equation 4.2.1 to construct the fractals we have covered. For example, we can use it to change the position of fractals in the x and y axis:

- To change the position of the fractal in the x-axis we change the value of a
- To change the position of the fractal in the y-axis we change the value of b

Moreover, we can change the position of these objects by reflection and rotation:

- To rotate a fractal through an angle θ anticlockwise by the origin, we can multiply S by the rotation matrix $Ro_\theta = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$
- To reflect a fractal through an angle θ anticlockwise by the x-axis, we multiply S by the reflection matrix $Re_\theta = \begin{bmatrix} \cos 2\theta & \sin 2\theta \\ \sin 2\theta & -\cos 2\theta \end{bmatrix}$

Finally, we can change the size of our fractal by multiplying our equation by c (the contraction factor).

We will now create an iterated function system for each of our examples covered in chapter 3. We will also explore a variation of Sierpinski's triangle – Sierpinski's carpet.

4.3. Sierpinski's Triangle

To construct this geometrical object in the (x,y)-plane, we require three different functions:

$$F_1(x,y) = \frac{1}{2}\begin{bmatrix} x \\ y \end{bmatrix} = \left(\frac{1}{2}x, \frac{1}{2}y\right) \quad (4.3.1)$$

$$F_2(x,y) = \frac{1}{2}\begin{bmatrix} x \\ y \end{bmatrix} + \frac{1}{2}\begin{bmatrix} 1 \\ 0 \end{bmatrix} = \left(\frac{1}{2}x + \frac{1}{2}, \frac{1}{2}y\right) \quad (4.3.2)$$

$$F_3(x,y) = \frac{1}{2}\begin{bmatrix} x \\ y \end{bmatrix} + \frac{1}{2}\begin{bmatrix} 1/2 \\ \sqrt{3}/2 \end{bmatrix} = \left(\frac{1}{2}x + \frac{1}{4}, \frac{1}{2}y + \frac{\sqrt{3}}{4}\right) \quad (4.3.3)^{[16]}$$

In equation 4.3.1, a = 0, b =0 and c = $\frac{1}{2}$. This means there is no change in position in the x-axis or y-axis, but the size of the object has been scaled by $\frac{1}{2}$. We can see that in all equations, our contraction factor remains the same. So the object is continuously being scaled by $\frac{1}{2}$.

Observing the next equation, 4.3.2, we can see that a is now equal to $\frac{1}{2}$, so the fractal has been moved by $\frac{1}{2}$ in the direction of the x axis. Our last mapping (4.3.3) shows a shift in the y-axis by $\frac{\sqrt{3}}{4}$.

These three transformations allow us to construct the Sierpinski triangle, as shown in fig 4.3.4. These can then be reproduced when iterating.

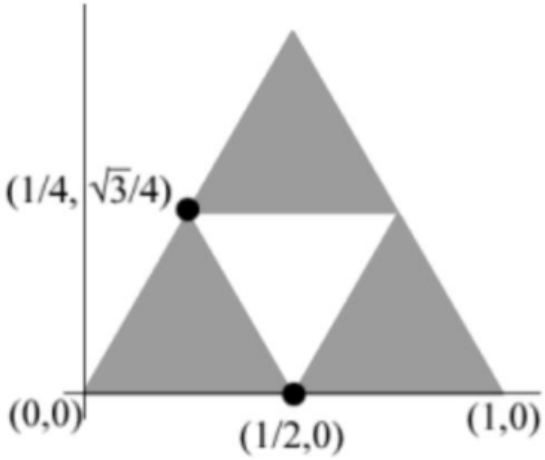

Fig 4.3.4: Iterated function system on Sierpinski's triangle[17]

4.4. Koch Curve

This fractal is created using scaling, rotation and reflection. Four different mappings are required to create its iterated function system. They are as follows:

$$F_1(x,y) = \frac{1}{3}\begin{bmatrix} x \\ y \end{bmatrix} = \left(\frac{1}{3}\,x,\ \frac{1}{3}\,y\right) \tag{4.4.1}$$

$$F_2(x,y) = \frac{1}{3}\begin{bmatrix} \cos\frac{\pi}{3} & -\sin\frac{\pi}{3} \\ \sin\frac{\pi}{3} & \cos\frac{\pi}{3} \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} + \frac{1}{3}\begin{bmatrix} 1 \\ 0 \end{bmatrix} = \left(\frac{1}{6}\,x - \frac{\sqrt{3}}{6}y + \frac{1}{3}\ ,\frac{\sqrt{3}}{6}x + \frac{1}{6}\right) \tag{4.4.2}$$

$$F_3(x,y) = \frac{1}{3}\begin{bmatrix} \cos\frac{2\pi}{3} & \sin\frac{2\pi}{3} \\ \sin\frac{2\pi}{3} & \cos\frac{2\pi}{3} \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} + \frac{1}{3}\begin{bmatrix} 2 \\ 0 \end{bmatrix} = \left(\frac{1}{6}\,x + \frac{\sqrt{3}}{6}y + \frac{1}{2}\ ,-\frac{\sqrt{3}}{6}x + \frac{1}{6}\,y + \frac{\sqrt{3}}{6}\right) \tag{4.4.3}$$

$$F_4(x,y) = \frac{1}{3}\begin{bmatrix} x \\ y \end{bmatrix} + \frac{1}{3}\begin{bmatrix} 2 \\ 0 \end{bmatrix} = \left(\frac{1}{3}\,x + \frac{2}{3},\frac{1}{3}\,y\right) \tag{4.4.4}$$

It is clear by equation 4.4.1 that our contraction factor is $\frac{1}{3}$. By using the rotation matrix to generate equation 4.4.2, we have rotated the fractal anticlockwise by 60°. We have also shifted the fractal by $\frac{1}{3}$ in the x axis.

The third contraction mapping introduces the reflection matrix. So the object has been reflected anticlockwise by the x- axis. It has also been shifted by $\frac{2}{3}$ in the direction of the x-axis.

Our last mapping is a repetition of equation 4.4.1, except part of the structure is shifted by $\frac{2}{3}$ in the direction of the x axis.

4.5. Cantor Set

Two transformations are needed to create the Cantor set. Using our general equation, we can obtain them:

$$F_1(x,y) = \frac{1}{3}\begin{bmatrix} x \\ y \end{bmatrix} = \left(\frac{1}{3}\,x, \frac{1}{3}\,y\right) \quad (4.5.1)$$

$$F_2(x,y) = \frac{1}{3}\begin{bmatrix} x \\ y \end{bmatrix} + \frac{1}{3}\begin{bmatrix} 2 \\ 0 \end{bmatrix} = \left(\frac{1}{3}\,x + \frac{2}{3}, \frac{1}{3}\,y\right) \quad (4.5.2)$$

This is quite simple in comparison to the other fractals we have worked with. We contract the object by $\mathbf{\frac{1}{3}}$ to obtain the first transformation. Finally, we take this mapping and shift it by $\mathbf{\frac{2}{3}}$ in the direction of the x-axis (equation 4.5.2).

4.6. Sierpinski's Carpet

This is a variation of Sierpinski's triangle. Thus its construction is very similar, the only difference being that we are dealing with squares.

To create this shape, we take a square and divide it into nine smaller equal squares. We then remove the middle square to create a hole. This leaves us with eight squares. We repeat this process for each of the squares. This creates the second iteration. Iterating the squares continuously creates the Sierpinski's carpet (fig 4.6.9).
Since we are shifting the square seven times, we need eight different equations:

$$\mathbf{F_1}(x,y) = \mathbf{\frac{1}{3}}\begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} = \left(\mathbf{\frac{1}{3}}\,\mathbf{x}, \mathbf{\frac{1}{3}}\,\mathbf{y}\right) \quad (4.6.1)$$

$$F_2(x,y) = \frac{1}{3}\begin{bmatrix} x \\ y \end{bmatrix} + \frac{1}{3}\begin{bmatrix} 1 \\ 0 \end{bmatrix} = \left(\frac{1}{3}\,x + \frac{1}{3}, \frac{1}{3}\,y\right) \quad (4.6.2)$$

$$F_3(x,y) = \frac{1}{3}\begin{bmatrix} x \\ y \end{bmatrix} + \frac{1}{3}\begin{bmatrix} 2 \\ 0 \end{bmatrix} = \left(\frac{1}{3}\,x + \frac{2}{3}, \frac{1}{3}\,y\right) \quad (4.6.3)$$

$$F_4(x,y) = \frac{1}{3}\begin{bmatrix} x \\ y \end{bmatrix} + \frac{1}{3}\begin{bmatrix} 0 \\ 1 \end{bmatrix} = \left(\frac{1}{3}\,x, \frac{1}{3}\,y + \frac{1}{3}\right) \quad (4.6.4)$$

$$F_5(x,y) = \frac{1}{3}\begin{bmatrix} x \\ y \end{bmatrix} + \frac{1}{3}\begin{bmatrix} 0 \\ 2 \end{bmatrix} = \left(\frac{1}{3}\,x, \frac{1}{3}\,y + \frac{2}{3}\right) \quad (4.6.5)$$

$$F_6(x,y) = \frac{1}{3}\begin{bmatrix} x \\ y \end{bmatrix} + \frac{1}{3}\begin{bmatrix} 1 \\ 2 \end{bmatrix} = \left(\frac{1}{3}\,x + \frac{1}{3}, \frac{1}{3}\,y + \frac{2}{3}\right) \quad (4.6.6)$$

$$F_7(x,y) = \frac{1}{3}\begin{bmatrix} x \\ y \end{bmatrix} + \frac{1}{3}\begin{bmatrix} 2 \\ 1 \end{bmatrix} = \left(\frac{1}{3}\,x + \frac{2}{3}, \frac{1}{3}\,y + \frac{1}{3}\right) \quad (4.6.7)$$

$$F_8(x,y) = \frac{1}{3}\begin{bmatrix} x \\ y \end{bmatrix} + \frac{1}{3}\begin{bmatrix} 2 \\ 2 \end{bmatrix} = \left(\frac{1}{3}\,x + \frac{2}{3}, \frac{1}{3}\,y + \frac{2}{3}\right) \quad (4.6.8)$$

By studying these mappings, it is clear that our contraction factor is $\frac{1}{3}$. Our first equation introduces this. This can also be picked up by our discussion, where we spoke about dividing the square into nine equal squares.

Our first transformation, explained by equation 4.6.1, shifts the object in the x axis by $\frac{1}{3}$. Subsequently, it is shifted again in the same direction by $\frac{2}{3}$. The same pattern that follows for the x axis also follows for the y axis. This is well-defined in equation 4.6.4 and 4.6.5 where the fractal is shifted by $\frac{1}{3}$ in the y axis, and then $\frac{2}{3}$ (respectively).

The last three equations show changes in both the x and y axis, at the same time. For example, equation 4.6.6 displays a shift in the x axis by $\frac{1}{3}$ and a shift in the y axis by $\frac{2}{3}$. It then switches, so in the sixth mapping it shows that the object has been shifted by $\frac{2}{3}$ in the x axis and $\frac{1}{3}$ in the y axis.

Finally, the last self-affined transformation (equation 4.6.8) displays a shift in the x and y axis by $\frac{2}{3}$.

We must take into account that throughout the different mappings, the contraction factor remains the same, so our object is continuously being scaled by $\frac{1}{3}$.

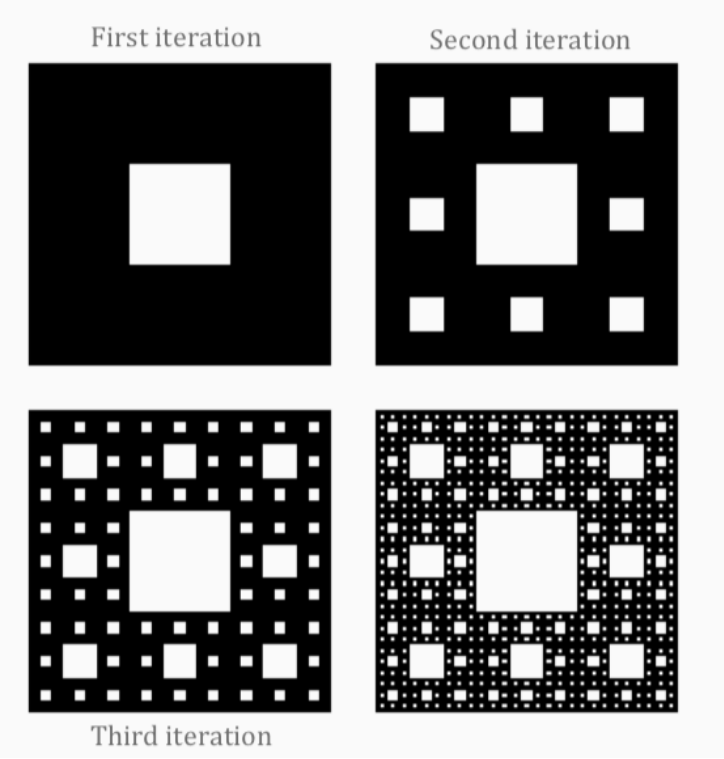

Fig 4.6.9: Sierpinski's Carpet[18]

4.7. Summary

By studying iterated function systems, we were able to discuss the different systems created for each fractal. It was interesting to see the variation in their complexities. For example, from studying the Koch curve we saw that it required the use of reflection and rotation matrices, as well as shifts in the both axes. On the other hand, the Cantor set only required two functions. These involved contractions in the x and y axes, and a shift in the x axis.

In addition to this, studying Sierpinski's carpet allowed us to explore its similarities against Sierpinski's triangle. In this iterated function system, the fractal was contracted in both axes. As the self-similar figure was continuously being shifted, eight different mappings were required to create this structure. Evidently, there were a greater number of squares so this object required far more mappings than Sierpinski's triangle.

In order to implement these systems, it would be useful to see how they can be used to visually display such fractals. This would be more technical, in the sense that we would consider using a programming software.

5. Conclusion

Throughout this essay we have been able to explore the nature of fractal geometry and properties that reside in this aspect of mathematics. By initially looking at the coastline, we were able to come to terms with the idea that lengths can be infinite. This motivated us to study the reasoning behind this.

Reading Benoit Mandelbrot's 'The Fractal Geometry of Nature' showed us the Richardson effect, which helped to explain this theory. From this, we were able to determine the dimension of these natural structures; and saw that they could be considered as fractals. This meant that we could use the Richardson effect more generally.

Studying the coastline introduced another notion – the box counting dimension. This proved as a useful tool in calculating fractal dimensions, and was used in a range of examples in chapter 3. Additionally, providing illustrations of our examples meant that we were able to analyse their self-similar nature. We also came to realise that the dimension of fractals did not exist as integers, but instead real numbers.

Finally, we considered how these structures could be constructed algebraically. Studying iterated function systems helped us to understand how fractals could be drawn on the (x,y)-plane. It was interesting to see the levels of complexity in creating iterated function systems, depending on the fractal structure.

Introducing Sierpinski's carpet was useful in looking at how its equations were similar yet different to Sierpinski's triangle. It made us curious in thinking about how the iterated function systems of other variations of our fractal examples may be, such as the Koch snowflake. In addition to this, constructing these recursive equations motivated us to think about how we could use these to create visual representations.

Advancing my technical capabilities would be beneficial in broadening my knowledge in generating fractals. For example, using a programming software (such as Maple) would enable me to implement our iterated function systems, to help visually represent these fractal structures.

Bibliography

[1] Kenneth Falconer. *Fractal Geometry: Mathematical Foundations and Applications, Chapter 9*. Third edition.

[2] Alex Clark. *Chaos and Fractals Lecture Notes, Chapter 6.* Queen Mary, University of London, 2020.

[3] Zahra Iqra Hussain. *Created using 'Shapes'*, 2020.

[4] Kenneth Falconer. *Fractal Geometry: Mathematical Foundations and Applications, Chapter 9*. Third edition.

[5] Andy Kiersz. *Business Insider: Fractals and the Coast of Great Britain,* 2019.

[6] Benoit Mandelbrot. *The Fractal Geometry of Nature, Chapter 5.* W.H. Freeman and Co, 1982.

[7] Andy Kiersz. *Business Insider: Fractals and the Coast of Great Britain,* 2019.

[8] Benoit Mandelbrot. *The Fractal Geometry of Nature, Chapter 5.* W.H. Freeman and Co, 1982.

[9] Benoit Mandelbrot. *The Fractal Geometry of Nature, Chapter 5.* W.H. Freeman and Co, 1982.

[10] Benoit Mandelbrot. *The Fractal Geometry of Nature, Chapter 5.* W.H. Freeman and Co, 1982.

[11] Allouche, J.-P. and Shallit, J. *Automatic Sequences: Theory, Applications, Generalizations.* Cambridge, England: Cambridge University Press, 2003.

[12] Julie. *Fractal Foundation: Koch Curve and Coastlines*, 2019.

[13] Thefrettinghand. *Cantor's Set, PNG*, 2007.

[14] Google Maps. *https://www.google.com/maps,* April 2020.

[15] Alex Clark. *Chaos and Fractals Lecture Notes, Chapter 6.* Queen Mary, University of London, 2020.

[16] Ciesielski, Krzysztof and Zdzislaw Pogoda. *"The Beginning of Polish Topology," The Mathematical Intelligencer*, 32-39, 1996.

[17] Devaney, Robert. *A First Course in Chaotic Dynamical Systems.* Addison-Wesley Publishing Co., 1992.

[18] https://www.technologyreview.com/2012/03/19/187121/infinity-computer-calculates-area-of-sierpinski- carpet- exactly/