

Implementation of Fractals in Maple

Marzena Szrama

Abstract

The project reviews some basic properties of fractals such as self-similarity and non integer dimension. The intention of this paper is to serve introduction to construction of the fractal structures. The programming tools in Maple are used to implement the Iterated Function Systems numerically, and to display the associated invariant sets.

1 Introduction.

Natural shapes like clouds, plants and coastlines are very irregular, so it is impossible to describe them using classical geometry. In this Essay I will use some properties of the so called Fractal Geometry, which was coined by Benoit B. Mandelbrot in the late 1970s. By considering concepts of dimension and self-similarity I will define fractals. To construct fractal structure I will study the underlying mathematical framework of Iterated Function Systems. Using these mathematical and numerical tools in maple I will display fractal objects. Finally, I will show how to compose own fractal pictures.

2 Dimension

Dimension is the number which geometrically describes objects. For example we plot statistical data on a 2-dimensional plane. However, in the everyday life the notion of dimension is quite an intuitive quantity.

2.1 Mathematical Concept of Dimension.


In Mathematics we often work in 3-dimensional space, the so called Euclidean space. For example, a point has dimension $D=0$, a line has $D=1$, a plane has $D=2$ and a sphere has $D=3$. So an object in Euclidian Geometry has dimension, which is a positive integer.

In the linear vector spaces a minimal spanning set of linearly independent vectors forms its basis. If a vector space V has a basis consisting of n vectors, then V has dimension n . For

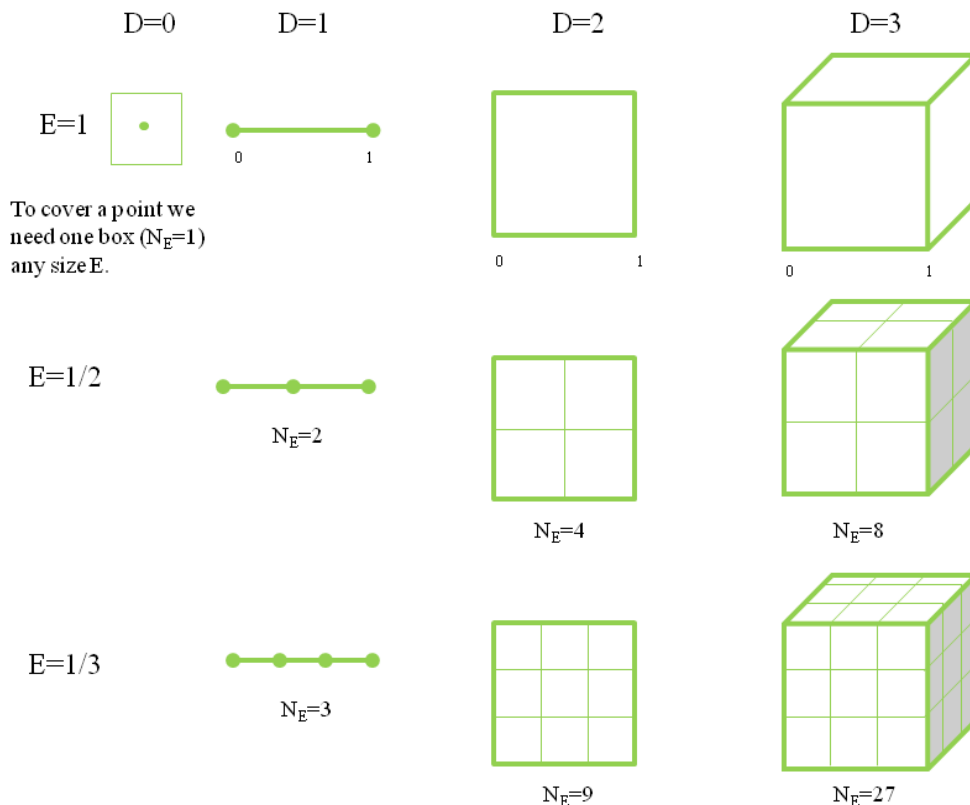
example $\{e_1, e_2, e_3\}$ forms the standard basis in \mathbb{R}^3 , where $e_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$, $e_2 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$, $e_3 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$,

so the number of vectors in a basis is 3, hence the dimension is $D=3$. Therefore the dimension $D=n$ is also an integer and it can be a finite or infinite number for n -dimensional vector spaces in \mathbb{R}^n . Since we can calculate the dimension for linear vector spaces, we want to do that for other objects.

2.2 Dimension via Covering for Euclidean Objects.

To illustrate dimension we relate it to the size of Euclidean objects. For a line we measure its length, for a square its area and for a cube its volume. We introduce the rectangle  with side length E . To find the size of the geometrical objects we cover them with these small boxes and count its number N_E .

We consider regular objects like: point, unit line interval, unit square and unit cube.



Figure(2.1): Box-Counting dimension for Euclidian objects.

The length of a line is:

$$A = EN_E \quad (1)$$

We can generalise the Equation(1) to the D dimensional case:

$$B = E^D N_E \quad (2)$$

From the Figure(2.1) we can notice the relation between the number of elements N_E and a side length E . For example to cover the unit square we can use one box size $E = 1$, or take four boxes size $E = 1/2$, then $N_E = 4 = 2^2 = (1/E)^D$, where D denotes the dimension of the measured object. Similarly if we use nine boxes size $E = 1/3$, we get $N_E = 9 = 3^2 = (1/E)^D$. The same argument works for a unit line and a unit cube. We can conclude that:

$$N_E = \frac{1}{E^D} = E^{-D} \quad (3)$$

Taking the logarithm of both sides;

$$\log N_E = \log \frac{1}{E^D} \quad (4)$$

Then:

$$D \log E^{-1} = \log N_E \quad (5)$$

Solving for D , we have:

$$D = \frac{\log N_E}{-\log E} \quad (6)$$

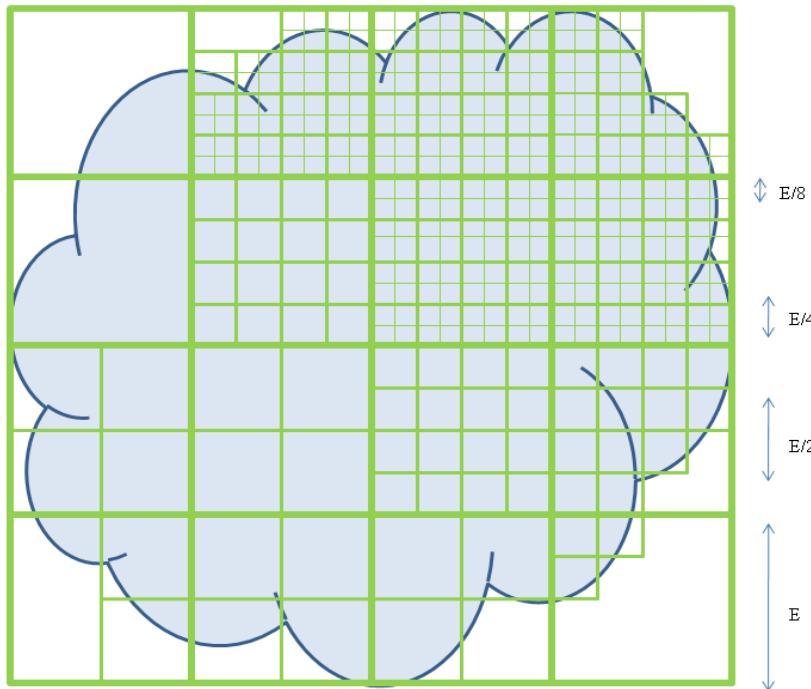
Letting E tend to zero, we get:

$$D = \lim_{E \rightarrow 0} - \frac{\log N_E}{\log E} \quad (7)$$

This is also called the Hausdorff – Besicovitch dimension.

2.3 Dimension via Covering for Natural Shapes.

The Box-Counting method of measurement can be used to describe the size of irregular objects. For example we take shape of a ‘cloud’. We cover the area by the rectangles (edge size E) as shown in the picture:



Figure(2.2): Box-Counting dimension for natural object.

We cover the shape with boxes to find how the smallest number of boxes changes with the size of boxes. In our measurement we can use any set size E (for example circle radius E). By reducing edge length E we increase the number of elements N_E to get more accurate, so we let E tend to zero. By the relation in Equation(7) we get approximate dimension D_{ap} by:

$$D_{ap} \simeq \lim_{E \rightarrow 0} -\frac{\log N_E}{\log E} \quad (8)$$

Solving Equation(2) for N_E and putting result in to Equation(8), we get:

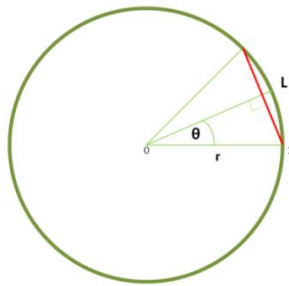
$$D_{ap} \simeq \lim_{E \rightarrow 0} -\frac{\log BE^{-D}}{\log E} = \lim_{E \rightarrow 0} \left(\frac{\log B}{-\log E} + \frac{-D \log E}{-\log E} \right) = 0 + D = D \quad (9)$$

We can conclude that estimated dimension D is independent of measuring unit E . Hence the Equation(8) gives us a good approximation of the dimension of natural objects (if the limit exists). This is also called the Kolmogorov Capacity¹ or the capacity dimension.

2.4 Describing the Size of Objects by Measuring Their Girth.

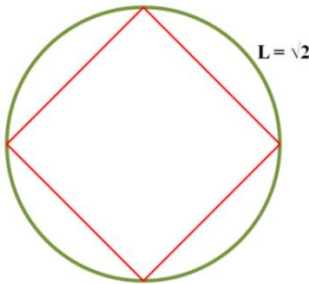
To illustrate this measurement we consider an example of a curve: a circle. We find the circumference C_k using a straight line interval of the length L , where k is the number of intervals used. We take a unit circle ($r = 1$) and calculate the value of L .

We proceed as in the next figure:

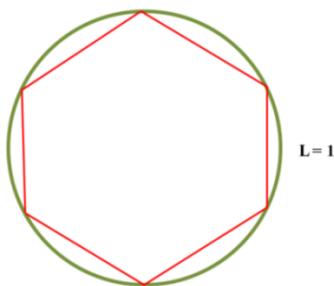


By the Pythagoras Theorem we get: $\sin \theta = \frac{1}{2}L$, then $L = 2 \sin \theta$.

Hence $C_k = kL$.



$\theta = \pi/4$, so $L = 2 \sin \pi/4 = \sqrt{2}$,
then $C_k = 4\sqrt{2}$, where $k = 4$



$\theta = \pi/6$, so $L = 2 \sin \pi/6 = 1$,
then $C_k = 6$, where $k = 6$

Figure(2.3): Measurement of the circumference by straight line interval.

¹ Hans Lauwerier, *Fractals Endlessly Repeated Geometrical Figures*, trans. by Sophia Gill-Hoffstadt (Princeton, New Jersey: Princeton University Press, 1991), p. 80.

We can conclude that:

$$C_k = kL \quad (10)$$

Where $L = 2\sin \theta$, and $\theta = \frac{\pi}{k}$

Next we want to relate k , L and the dimension. Using the Equation(6), where in this case k is the number of elements N_E and the length of a line intervals L is E , we get:

$$D = \frac{\log k}{-\log L} \quad (11)$$

To improve the accuracy we decrease the angle θ , so we let L tend to zero, as is shown in the Figure(2.3). Since L tends to zero, k tends to infinity, so by the Equation(7) we get:

$$D = \lim_{k \rightarrow \infty} \frac{-\log k}{\log 2\sin \frac{\pi}{k}} \quad (12)$$

We can use L'Hopital's rule, which says that we can find the limit of the ratio of two functions by finding the limit of the ratio of the derivatives of the numerator and denominator of these functions. Hence by L'Hopital's rule we have:

$$D = \lim_{k \rightarrow \infty} \frac{-\log k}{\log 2\sin \frac{\pi}{k}} = \lim_{k \rightarrow \infty} \frac{-\frac{1}{k}}{\frac{-\frac{\pi}{k^2} 2\cos \frac{\pi}{k}}{2\sin \frac{\pi}{k}}} = \lim_{k \rightarrow \infty} \frac{\sin \frac{\pi}{k}}{\frac{\pi}{k} \cos \frac{\pi}{k}} = \lim_{k \rightarrow \infty} \frac{\sin \frac{\pi}{k}}{\frac{\pi}{k}} \frac{1}{\cos \frac{\pi}{k}} \quad (13)$$

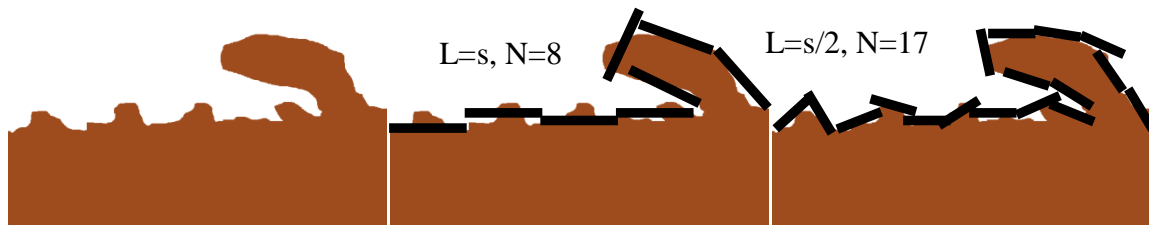
Using properties of the limit and L'Hopital's rule we get:

$$D = \lim_{k \rightarrow \infty} \frac{\sin \frac{\pi}{k}}{\frac{\pi}{k}} \lim_{k \rightarrow \infty} \frac{1}{\cos \frac{\pi}{k}} = 1 \lim_{k \rightarrow \infty} \frac{-\frac{\pi}{k^2} \cos \frac{\pi}{k}}{-\frac{\pi}{k^2}} = 1 \quad (14)$$

The dimension of the circle is 1, so the Equation(7) is satisfied. Hence the definition of dimension derived from the Box-Counting is valid for measuring circumference.

2.5 Measurement of Coastline Length.

The above method can be used in the measurement of the length of more complicated objects. For example we can use stick size L to estimate the length of a coastline C . We proceed as in the picture:



Figure(2.4): Measurement of the coastline length.

The estimated length C is equal to the length of the ruler L multiplied by N_L , the number of such rulers needed to cover the measured object, see the Equation(1).

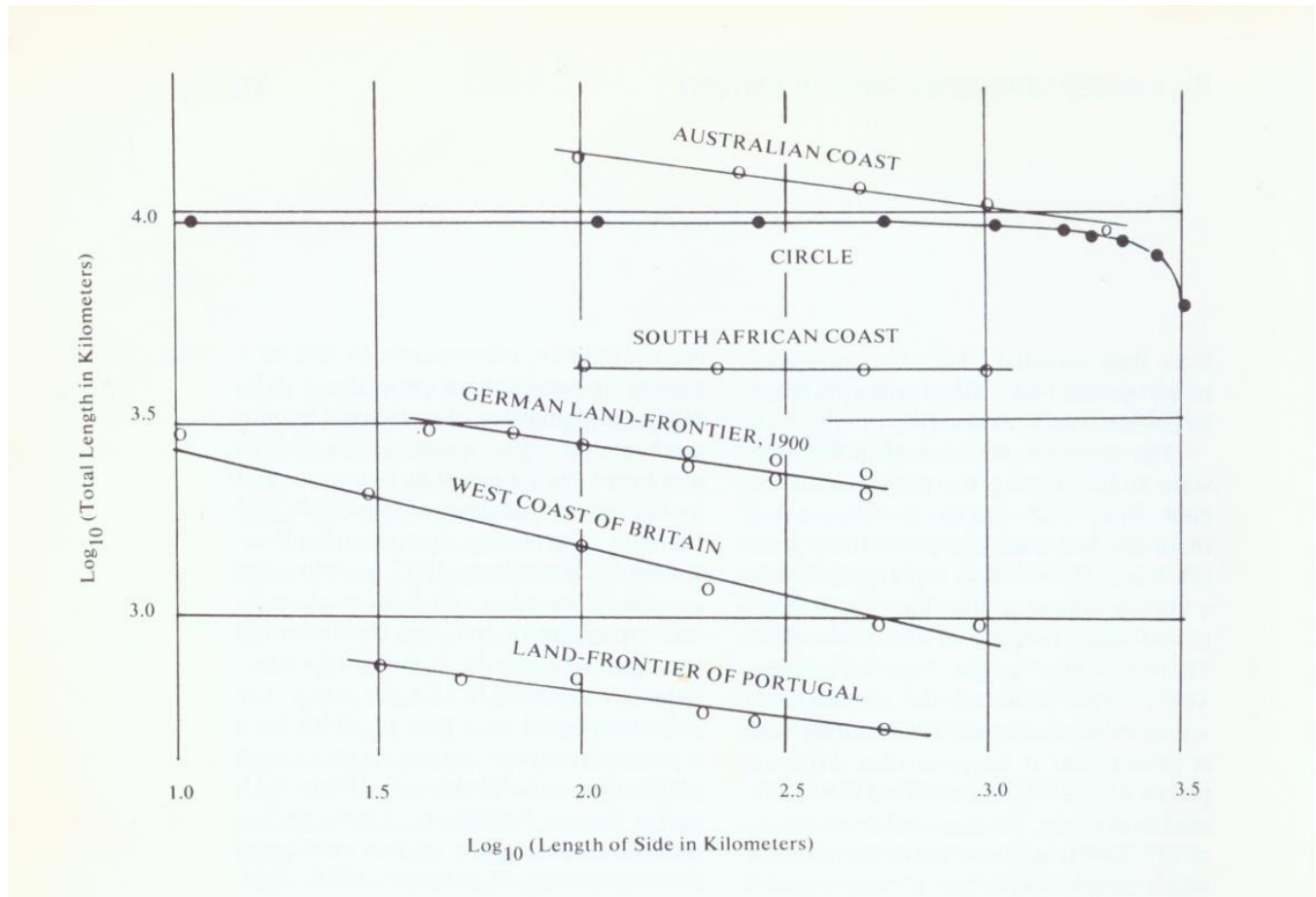
- Firstly we use ruler length $L=s$, where $C= N_L L = 8L = 8s$,
- Secondly, for $L = s/2$ we get $C = 17L = N_L L = 17s/2 = 8.5s$.

So if we take smaller and smaller ruler, the length of the coastline gets larger and larger.

Hence if we let L tend to zero, the length will increase. Therefore we can conclude that, by decreasing size of the ruler we will get an infinitely long coastline. But what is its dimension?

2.6 Richardson Effect.

Benoit B. Mandelbrot began his book *Fractals from chance, and dimension* by considering the question: How long is the coast of Britain? Using Richardson's experimental measurement of coastlines lengths we can analyse his results on the diagram in the Figure(2.5). Lewis Fry Richardson first noted the regularity between the length of national boundaries and the scale size. The relation between estimated length and the length of scale is linear with negative slope on the logarithmic plot. This is the so called Richardson Effect.



Figure(2.5): Richardson's empirical data on the rate of increase of coastlines lengths.²

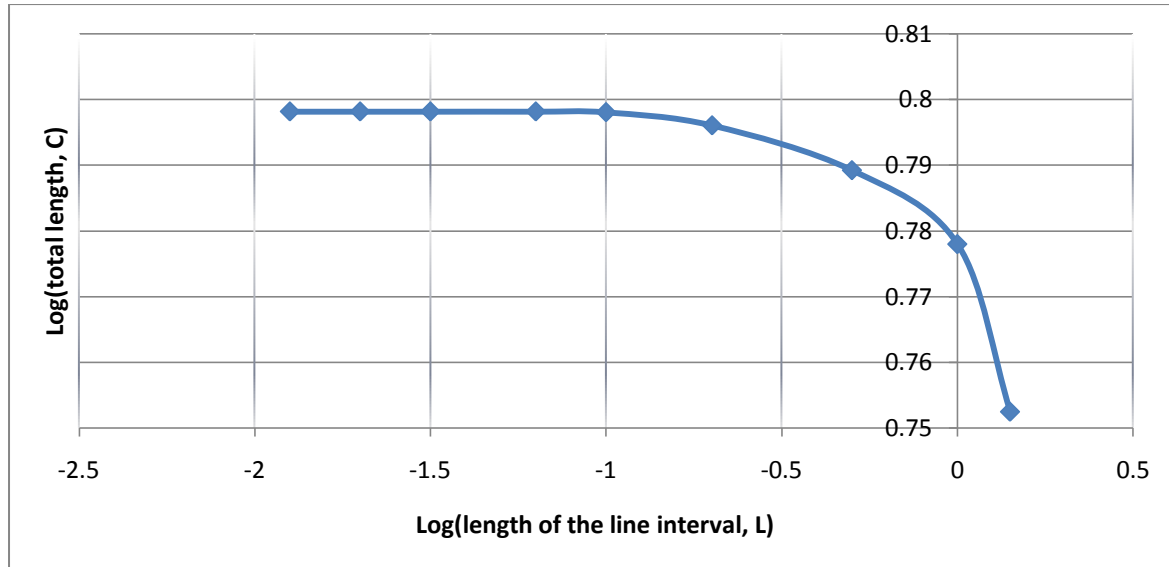
We can notice from the Figure(2.5) that the slope for the circle is zero when the side length is less than 3km. So we can assume that the measurement by a ruler length less than 3000 m is accurate. The slope for the South Africa is also close to zero since its coastline is relatively 'smooth' comparing to the West Britain coastline, which is very rough. Hence these slopes are less than zero and decrease proportionally to the 'roughens' of the measured coastline. Therefore we can conclude that if a line on the logarithmic plot is steeper, then the resulting length of the coastline increase more rapidly. Slopes S for four other coastlines are negative and represent different numbers from the set $(-1,0)$. B.B. Mandelbrot concluded that these slopes are related to dimension by: $S = (1 - D)$. Therefore the dimension of the coastlines has the value between one and two.

² B. Benoit Mandelbrot, *The Fractal Geometry of Nature*, (San Francisco: W. H. Freeman and Company, 1977), p. 32.

L. F. Richardson analysing diagrams in the Figure(2.5) concluded that the number of elements needed to measure the coastline length is approximately $N_E \simeq \lambda E^{1-D}$, where λ is a constant.³ Taking logarithms of both sides of this equation we get:

$$\log N_E \simeq \log \lambda E^{1-D} \simeq \log \lambda + \log E^{1-D} \simeq (1-D)\log E + \log \lambda \simeq \text{Slog } E + \log \lambda \quad (15)$$

The Equation(15) can be used to produce the log-log plot, where λ denotes constant specific to each coastline. We produce such plot for circumference in previous section, Figure(2.3).



Figure(2.6): Logarithmic plot for unit circle with $\lambda = 1$.

We can notice that in the above plot is similar to a result on the diagram in the Figure(2.5). Hence by approximating the length of connected physical curves we can estimate their dimension.

2.7 Conclusion.

The concept of dimension has different meanings in the literature. In mathematics we have different definitions of dimension. In this essay we consider the Box-Counting Dimension. The Equation(7) defines the dimension as a fraction of logarithms of the number of boxes and their size. Since a ratio of two real numbers is involved, we can conclude that the dimension D does not need to be an integer. It can be a real number.

Using Richardson's Effect we can conclude that the dimension of coastlines is not an integer. This leads to the conclusion: **an object whose dimension is not an integer is said to be a fractal.** The fractal dimension can be defined by using real world data, and can be measured approximately by experiments. Hence the coastlines curves are examples of fractals, with fractal dimension between the dimensionality of a line and a plane.

What more can we say about fractals?

³ B. Benoit Mandelbrot, *The Fractal Geometry of Nature*, (San Francisco: W. H. Freeman and Company, 1977), p. 31.

3 Self-Similarity

A self-similar object has the same shape as its parts after rescaling. Hence self-similar objects look exactly or approximately like their parts.

How we can relate self-similarity to fractals?

3.1 Coastlines.

We consider our first example of a fractal by looking at the satellite view from a google map.

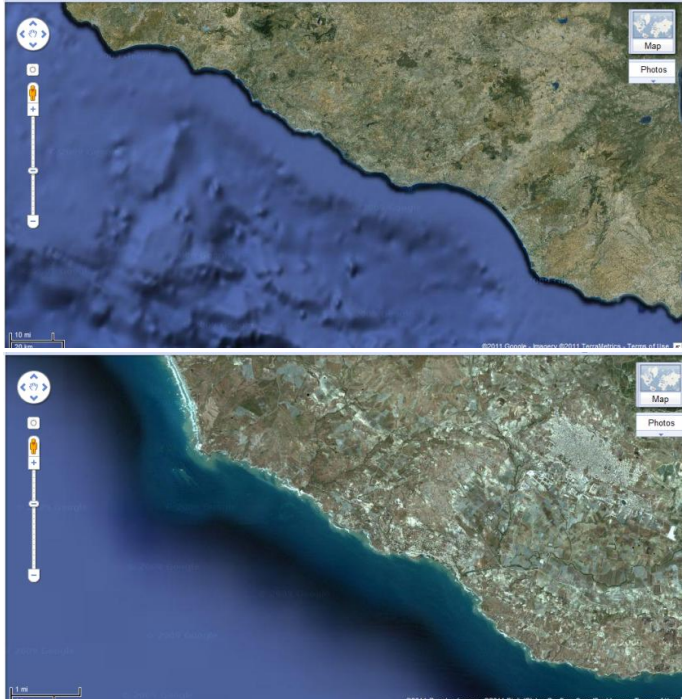
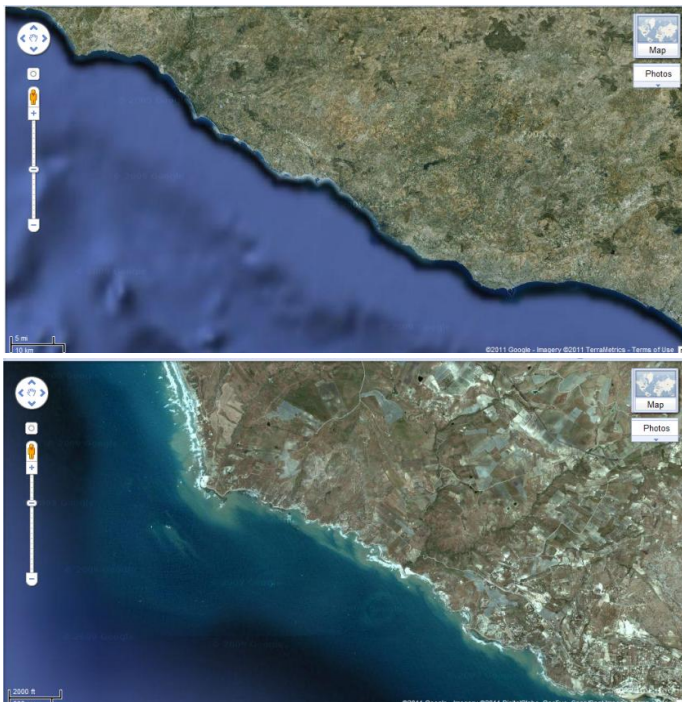


Figure (3.1): Sicily coastline picture made from 20km and 1km distance.⁴



Figure(3.2): Sicily coastline picture made from 10km and 500m distance.

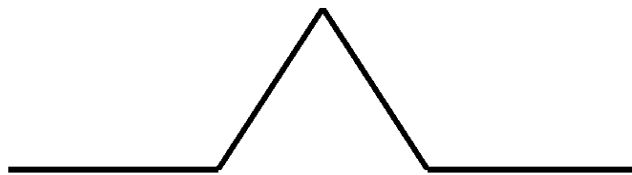
⁴Google Maps', <<http://maps.google.com/>>, [accessed 20 March 2011] .

From above figures we can notice that the coastline looks almost the same on different scales. By zooming in the view, we get its smaller fragments. If we look at two pictures of the boarder curve we are not able to recognise distances. Parts of the above coastline are very similar on the ever repeating scale. Hence fragments of the coastline are identical in distribution to each other. Therefore we can assume that coastlines are statistically self-similar.

Is it possible to construct such objects?

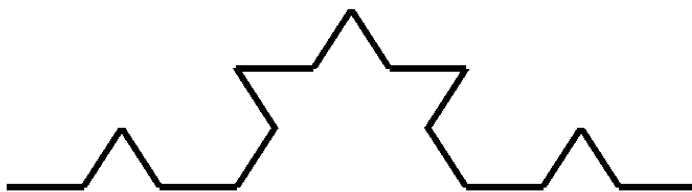
3.2 Koch Curve.

The mathematical model of a coastline can be represented by the Koch Curve. To construct this curve we start with a line interval length 1. Next we replace the middle third of a line by two lines, each having the same length $1/3$ as two remaining lines on each side. We get a curve with length $L = 4/3$ (Figure(3.3)).



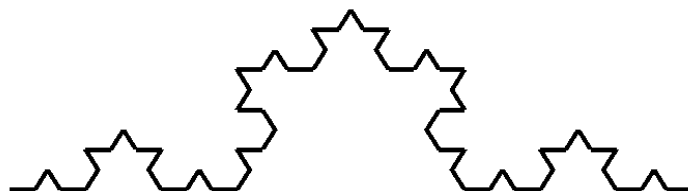
Figure(3.3): First iteration of the Koch Curve.

Then we repeat the procedure by replacing every line segment by 2 lines, each $1/3$ of the length of the original. We get a curve with the length $L = 16/9$ (Figure(3.4)).



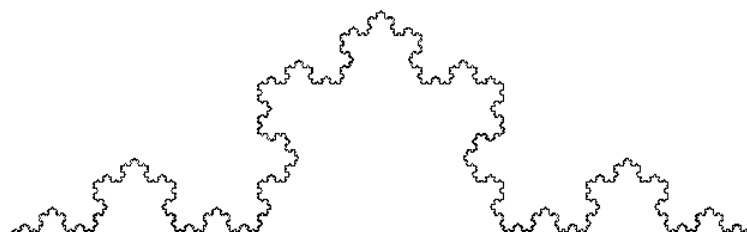
Figure(3.4): Second iteration of the Koch Curve.

In the next iteration we get a curve with $L=64/27$, so the length increases (Figure(3.5)).



Figure(3.5): Third iteration of the Koch Curve.

Hence after an infinite number of iterations we get an object with infinite length.

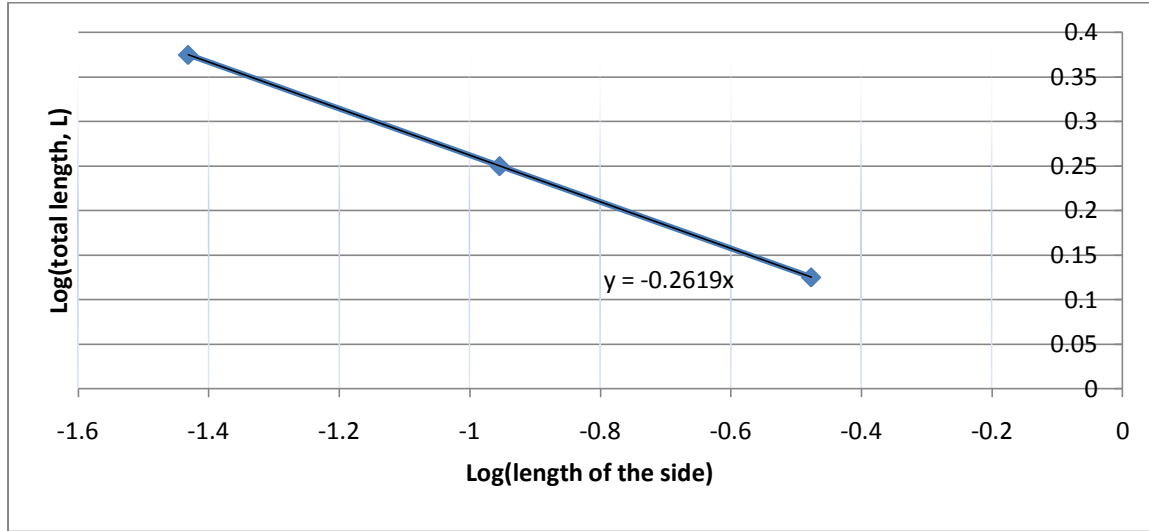


Figure(3.6): Koch Curve.

To calculate the dimension of such an object we need 4 squares ($N_E = 4$) edge size $E = 1/3$, so by the Equation(7) we get:

$$D = \lim_{E \rightarrow 0} \frac{\log N_E}{-\log E} = \frac{\log 4}{-\log \frac{1}{3}} = \frac{\log 4}{\log 3} \approx 1.26 \quad (16)$$

Therefore we get an object with the fractal dimension $D \approx 1.26$ which is between dimensionality of a line, 1 and a plane, 2. The Koch Curve is exactly self-similar since its every part has the same shape as the whole curve. Using the Equation(15) we can produce the log-log plot for this figure. Let, for simplicity $\lambda = 1$. We use values of lengths L calculated in above iterations of the Koch Curve.

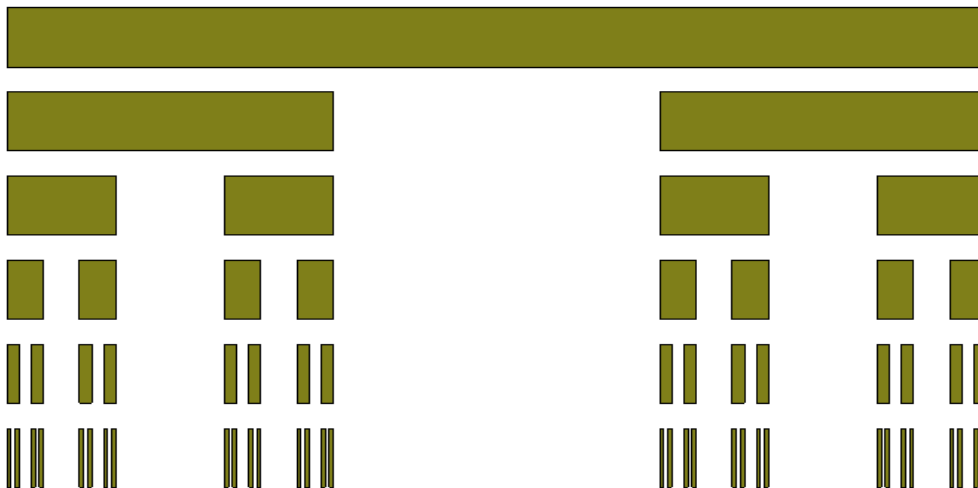


Figure(3.7): Logarithmic plot for the Koch Curve.

From the above figure we get the slope of a line $S \approx -0.2619$. Hence the dimension is $D \approx 1 + 0.2619 \approx 1.2619$, which gives the same result as in the Equation(16).

3.3 Cantor Set.

To illustrate the Cantor Set we take a unit length polygon and divide it into three equal parts. Next we remove the middle part, so we get the same two polygons with reduced size. Then we repeat the procedure for these objects. Hence we get $4=2^2$, $8=2^3$, $16=2^4$, $32=2^5$, $64=2^6$, ... objects in each iteration. The process is illustrated in the picture bellow:



Figure(3.8): Cantor Set.

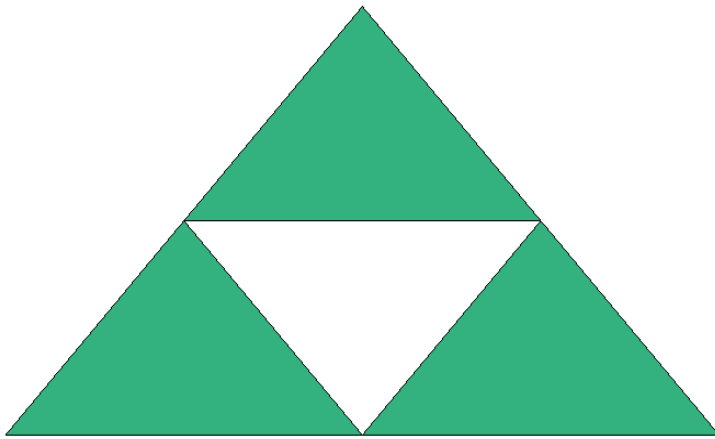
To calculate the dimension of the Cantor Set we need 2 squares ($N_E = 2$) edge size $E = 1/3$, hence using the Equation(7) we get:

$$D = \lim_{E \rightarrow 0} \frac{\log N_E}{-\log E} = \frac{\log 2}{-\log \frac{1}{3}} = \frac{\log 2}{\log 3} \simeq 0.63 \quad (17)$$

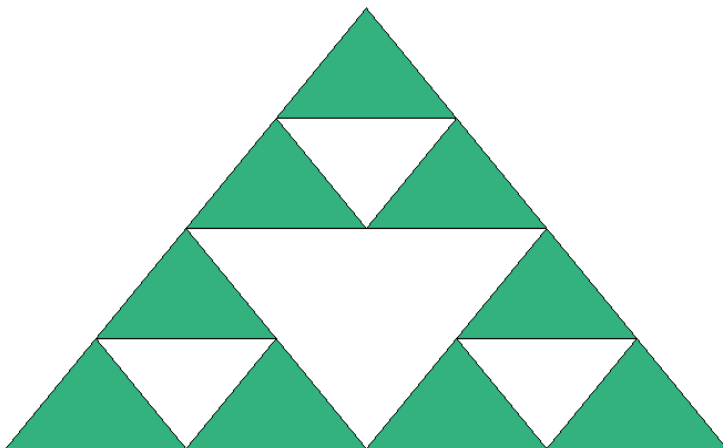
Therefore we get an object with the fractal dimension $D \simeq 0.63$ which is between dimensionality of a point, 0 and a line, 1. The Cantor Set is exactly self-similar on an ever repeating scale. By performing infinite number of iterations we will get a set containing infinitely many disjointed points (polygons). This is the so called Cantor Dust.

3.4 Sierpinski's Sieve.

To construct the Sierpinski's Sieve we take a triangle with equal side length of one unit ($E=1$), and we divide it into four equilateral triangles. Next we remove the middle one to make a triangle hole. So we have three smaller triangles, next we repeat the above procedure for these objects. This process is illustrated in the following figures:

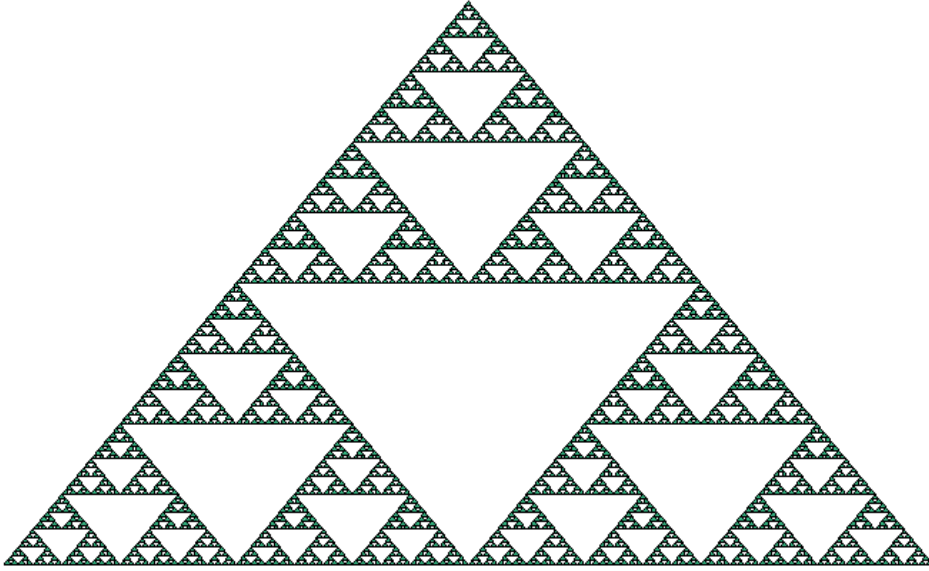


Figure(3.9): First iteration of Sierpinski's Triangle.



Figure(3.10): Second iteration of Sierpinski's Triangle.

In every step we remove parts of the figure, so we get more and more triangle holes. Hence the area of the whole triangle is decreasing, so after infinitely many iterations it is smaller and smaller (closer to zero).



Figure(3.11): Sierpinski's Sieve (Triangle).

To calculate the dimension of such an object we need 3 squares ($N_E = 3$) edge size $E = 1/2$, so by the Equation(7) we get:

$$D = \lim_{E \rightarrow 0} \frac{\log N_E}{-\log E} = \frac{\log 3}{-\log \frac{1}{2}} = \frac{\log 3}{\log 2} \approx 1.58 \quad (18)$$

By the Equation (7) the dimension of this figure is: $D \approx 1.58$, which is bigger than the dimension of the Koch Curve ($D \approx 1.26$). Hence the dimension of the Sierpinski's Triangle is between a line and a plane, but 'closer' to the dimension of a plane. This object is constructed by use of a repeating motif of the equilateral triangle, therefore it is exactly self-similar to every one of its parts. In each iteration we remove one third of the figure. Therefore the Sierpinski's Sieve is strictly self-similar object with a vanishing area.

3.5 Conclusion.

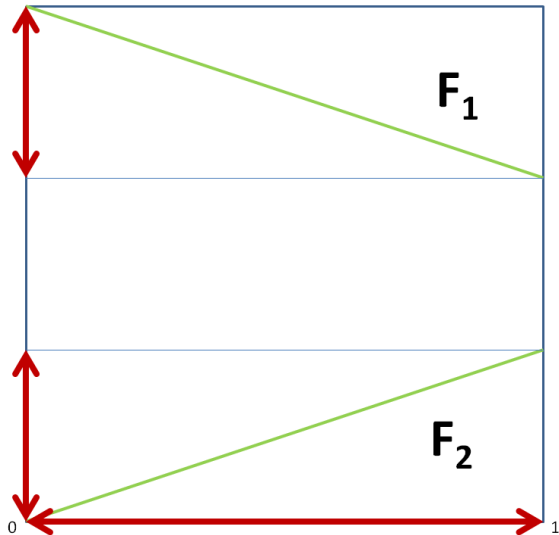
We can conclude that self-similarity is a property of fractals. These objects can be exactly or statistically self-similar. We have seen that coastlines are statistically similar to their parts on an ever repeating scale. The infinitely long Koch Curve is a scaled-down copy of the entire figure. The Cantor Set, made up of discrete points is exactly similar to its subsets. The Sierpinski's Sieve is made of smaller and smaller triangles which are perfectly similar to each other. Therefore **we define fractals as objects with built-in self-similarity**. These figures are made by use of simple geometrical transformations. In the next chapter we will discuss how to generate these repeating patterns.

4 Iterated Function Systems.

By examples of fractals in the previous chapter we can notice that in the first iterations of these figures are made of union of m rescaled copies of the initial set. The Cantor Set has $m=2$ (Figure (3.8)), the Sierpinski's Triangle has $m=3$ (Figure (3.11)), the Koch Curve has $m=4$ (Figure (3.6)). Hence fractals are constructed by m transformations, let call them F_m .

4.1 Similarity Transformations for the Cantor Set.

The Cantor Set is constructed by application of two functions to an initial set, for example a unit line interval. This is shown on the next picture:

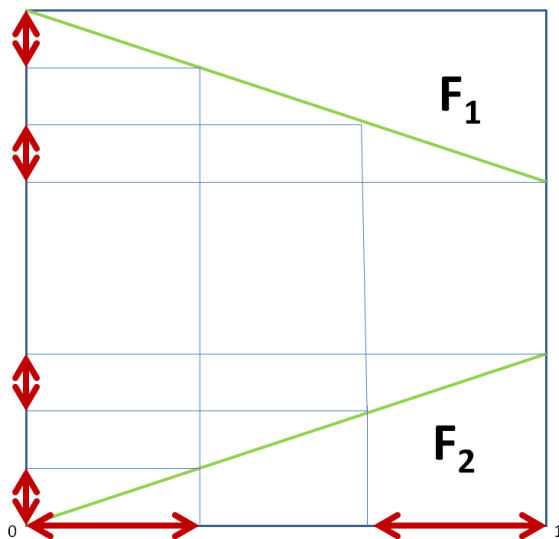


Figure(4.1): First Iteration of the Cantor Set.

Let call the initial set D_0 and the resulting set D_1 . By applying functions to a line interval we get two shrunk intervals. Hence we get:

$$D_1 = F_1(D_0) \cup F_2(D_0) \quad (19)$$

In the next iteration step we apply functions F_1 and F_2 to set D_1 , as in the picture below:



Figure(4.2): Second Iteration of the Cantor Set.

We call the resulting set D_2 , which is build by application of two functions to the set D_1 . Hence we get:

$$D_2 = F_1(D_1) \cup F_2(D_1) \quad (20)$$

We can generalise the above process to the n -th iteration step by:

$$D_{n+1} = F_1(D_n) \cup F_2(D_n) \quad (21)$$

Next letting n tend to infinity (we perform infinitely many iteration steps) we get:

$$D_\infty = F_1(D_\infty) \cup F_2(D_\infty) \quad (22)$$

So the resulting set is the limit set as the number of iterations n tends to infinity. Hence in each iteration step we get another set which is a subset of the initial set. Therefore we can conclude that the Cantor Set is invariant under mappings F_1 and F_2 .

4.2 Iterated Function Systems.

Let S be a closed subset of \mathbb{R}^n . The transformation $F_m: S \rightarrow S$ is called contraction mapping if there exists a number c such that $|F_m(x) - F_m(y)| \leq c|x - y|$ for all points x, y in S , where $c \in (0, 1)$ is contraction factor.

The mapping is called similarity transformation if $|F_m(x) - F_m(y)| = c|x - y|$ for all points x, y in S . This means that, if we apply contraction F_m to some object (closed set) we get the same object multiplied by factor c . Since c is less than one, an object is contracted. Let $F_1, F_2, F_3, \dots, F_m$ be contractions. We call a set D invariant under transformations F_m if:

$$F(D) = \bigcup_{i=1}^m F_i(D) \quad (23)$$

We define D as the subset of S and m as a finite natural number.⁵

The Iterated Function Systems (IFS) are families of contraction mappings which define closed invariant sets, which are often fractals.

An invariance means that if we apply the function to a point p in the set (fractal), we get a point p' which also belong to this set. Hence by use of IFS we can modify objects (sets) without changing their structure. Therefore using similarity transformations we can construct fractals. In the next chapter we will discuss this process.

⁵ Kenneth Falconer, *Fractal Geometry Mathematical Foundations and Applications*, (Chichester, England: John Wiley & Sons, 1990), p.113.

5 Construction of Fractals.

To construct fractal structures first we need to find their Iterated Function System.

5.1 Contraction Mappings in (x-y)-plane.

To classify the contraction mappings we consider linear functions $F_m: \mathbb{R}^2 \rightarrow \mathbb{R}^2$. Functions F_m are applied to the vector $D = \begin{bmatrix} x \\ y \end{bmatrix}$ to get the transformed vector $D' = \begin{bmatrix} x' \\ y' \end{bmatrix}$. Hence mappings F_m are applied to every element of the initial vector. Let D describe Cartesian coordinates of the point in the (x-y)-plane. Hence a list D' denotes transformed coordinates. We can generalist these transformations by:

$$D' = \begin{bmatrix} x' \\ y' \end{bmatrix} = F_m(D) = c \begin{bmatrix} x \\ y \end{bmatrix} + c \begin{bmatrix} a \\ b \end{bmatrix} = \begin{cases} cx + ca \\ cy + cb \end{cases} \quad (24)$$

Where a, b, c are constants and $\begin{bmatrix} a \\ b \end{bmatrix}$ is a vector. The objects generated by mappings F_m lie in the (x-y)-plane, where lists denote location of its points. Hence every point in the initial set (object) is represented in the (x-y)-plane by a vector D . Therefore we can change position of these figures, rotate them, reflect then by line through the origin and change their size:

- to change the position of an object along the x-axis we change the value of a ,
- to move an object along the y-axis we change the value of b ,
- to rotate an object anticlockwise through an angle θ about the origin, we multiply D from the right side by the rotation matrix:

$$A_\theta = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \quad (25)$$

- to reflect an object anticlockwise through an angle θ from the x-axis we multiply D from the right side by the reflection matrix:

$$B_\theta = \begin{bmatrix} \cos 2\theta & \sin 2\theta \\ \sin 2\theta & -\cos 2\theta \end{bmatrix} \quad (26)$$

- to change the size of an object we multiply the whole equation by the contraction factor c .

The function $F_m(D)$ is self-similar if the vector $\begin{bmatrix} a \\ b \end{bmatrix}$ is a zero vector, otherwise it is said to be self-affine. We call these transformations self-similar or self-affine respectively.

5.2 Construction of the Cantor Set.

To build the Cantor Set we need two similarity transformations, so by the Equation(24) we get:

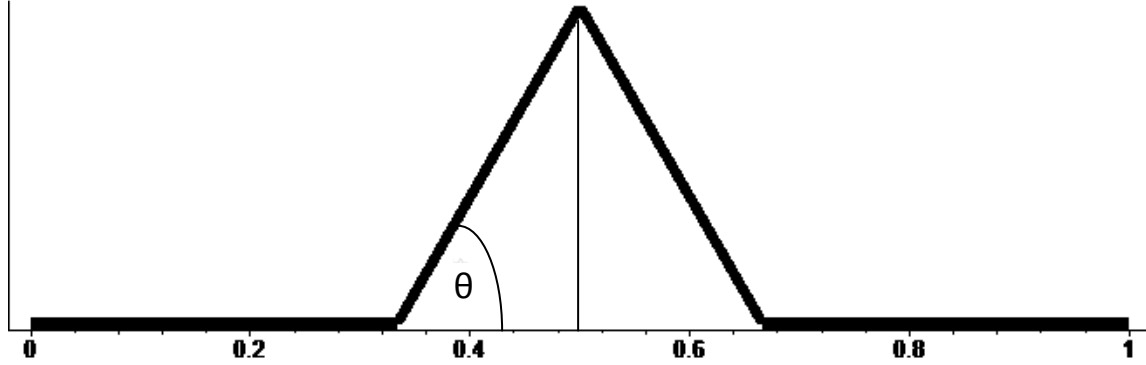
$$F_1(x, y) = \frac{1}{3} \begin{bmatrix} x \\ y \end{bmatrix} \quad (27)$$

$$F_2(x, y) = \frac{1}{3} \begin{bmatrix} x \\ y \end{bmatrix} + \frac{1}{3} \begin{bmatrix} 2 \\ 0 \end{bmatrix} \quad (28)$$

Where $a = 2$ is a shift in the x-axis direction, $b = 0$ denotes no shift in the y-axis direction, and $c = \frac{1}{3}$ is the contraction factor.

5.3 Construction of the Koch Curve.

This object is made of four similarity mappings: contraction by factor $1/3$, followed by rotation and reflection by angle the $\theta = \frac{\pi}{3}$ with displacement along the x-axis.



Figure(5.1): Construction of the Koch Curve

Hence by the Equation(24) we get four similarity mappings:

$$F_1(x, y) = \frac{1}{3} \begin{bmatrix} x \\ y \end{bmatrix} \quad (29)$$

$$F_2(x, y) = \frac{1}{3} \begin{bmatrix} \cos \frac{\pi}{3} & -\sin \frac{\pi}{3} \\ \sin \frac{\pi}{3} & \cos \frac{\pi}{3} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \frac{1}{3} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad (30)$$

$$F_3(x, y) = \frac{1}{3} \begin{bmatrix} \cos \frac{2\pi}{3} & \sin \frac{2\pi}{3} \\ \sin \frac{2\pi}{3} & -\cos \frac{2\pi}{3} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \frac{1}{3} \begin{bmatrix} 2 \\ 0 \end{bmatrix} \quad (31)$$

$$F_4(x, y) = \frac{1}{3} \begin{bmatrix} x \\ y \end{bmatrix} + \frac{1}{3} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad (32)$$

Can we combine Koch Curves to make new objects?

The answer is yes, since we can move curves in the (x-y)-plane using functions $F_m(D)$. For example we can take three Koch Curves and connect them to make the equilateral triangle.

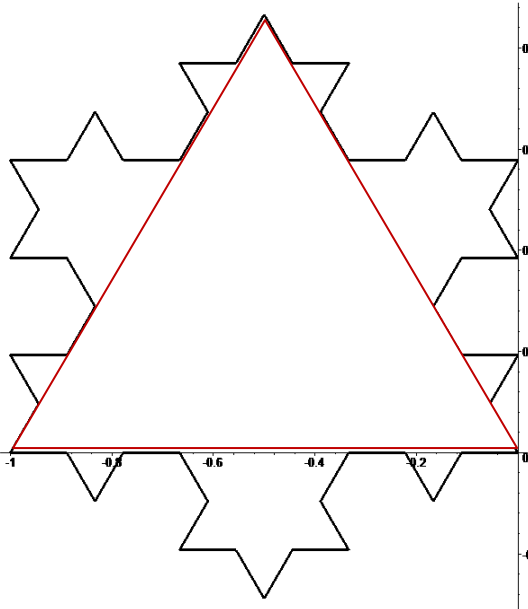
We construct the new figure, closed curve. To do that, we perform three reflections.

We need three mappings (Equation(24)):

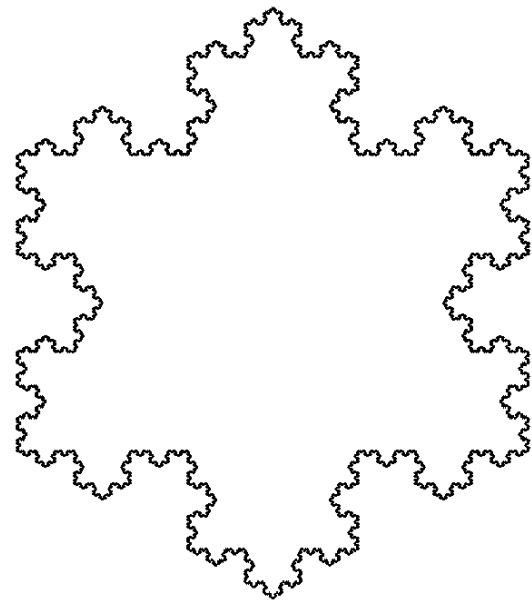
$$F_5(x, y) = - \begin{bmatrix} x \\ y \end{bmatrix} \quad (33)$$

$$F_6(x, y) = \begin{bmatrix} \cos \frac{-2\pi}{3} & \sin \frac{-2\pi}{3} \\ \sin \frac{-2\pi}{3} & -\cos \frac{-2\pi}{3} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} -1 \\ 0 \end{bmatrix} \quad (34)$$

$$F_7(x, y) = \begin{bmatrix} \cos \frac{2\pi}{3} & \sin \frac{2\pi}{3} \\ \sin \frac{2\pi}{3} & -\cos \frac{2\pi}{3} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \quad (35)$$



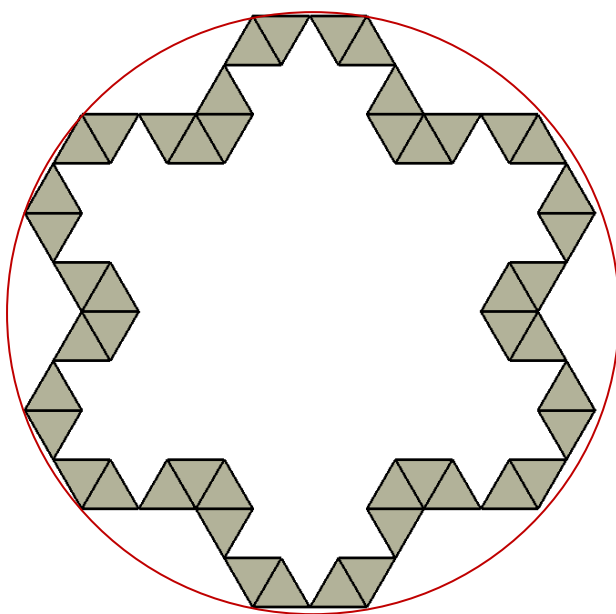
Figure(5.2): Construction of the Koch Island.



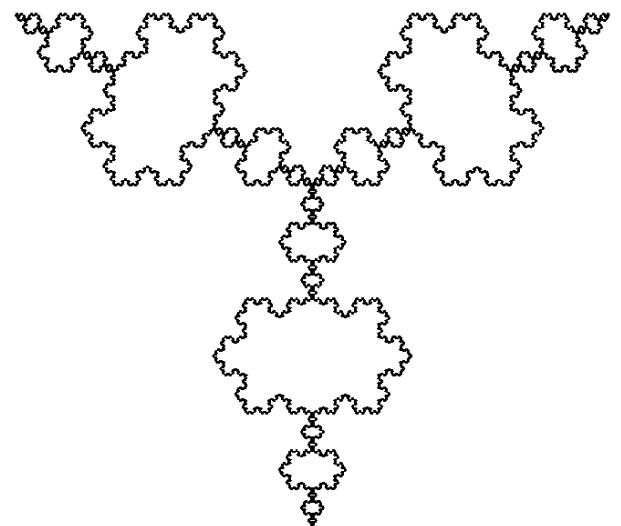
Figure(5.3): Koch Island (Snow Flake).

The Koch Island is the figure made up of fractal-curves. It is exactly self-similar object since its every part is a fractal. So itself it is a fractal figure with infinitely long circumference. We can notice that if we draw polygons around the coast of the figure it encloses the area. We can approximate it to the area of a circle. Hence we build the self-similar object with infinite length surrounding a finite area (Figure(5.4)).

Construction of the Koch Island involves two families of transformations. The first one contains one self-affine and three self-similar transformations to build the fractal curve. The second one involves three functions (without contraction) to build fractal figure. We can modify these mappings to get another version of the Koch Island. For example if we swap the angles in Equations: (34) and (35) we get the fractal figure folding inside out (Figure(5.5)).



Figure(5.4): The Koch Island enclosed in circle.



Figure(5.5): Variant of the Koch Curve.

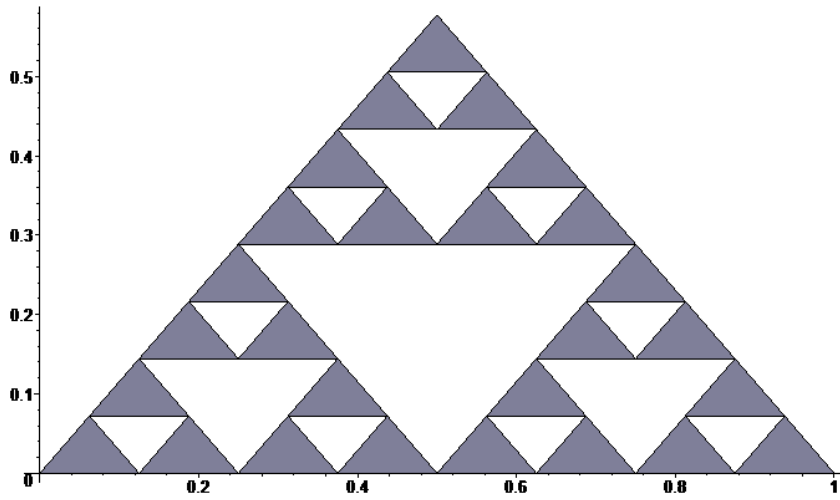
5.4 Construction of Sierpinski's Triangle.

The Sierpinski's Sieve is build by compression of equatorial triangle and two shifts in (x-y)-plane. To construct such object we need three transformations:

$$F_1(x, y) = \frac{1}{2} \begin{bmatrix} x \\ y \end{bmatrix} \quad (36)$$

$$F_2(x, y) = \frac{1}{2} \begin{bmatrix} x \\ y \end{bmatrix} + \frac{1}{2} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad (37)$$

$$F_3(x, y) = \frac{1}{2} \begin{bmatrix} x \\ y \end{bmatrix} + \frac{1}{2} \begin{bmatrix} \frac{1}{2} \\ \frac{\sqrt{3}}{3} \end{bmatrix} \quad (38)$$



Figure(5.6): Construction of the Sierpinski's Sieve.

5.5 Sierpinski's Carpet

Different variant of Sierpinski's figure is the Sierpinski's Carpet. This object is constructed by contracting and shifting unit square, Figure(5.7). We divide initial square into nine equal squares, so that the contraction factor is $c = 1/3$. Next we remove middle one to make square hole. This is performed by shrinking and moving square in the (x-y)-plane seven times, hence we need eight similarity mappings: one self-similar and seven self-affine.

By the Equation(24) we get:

$$F_1(x, y) = \frac{1}{3} \begin{bmatrix} x \\ y \end{bmatrix} \quad (39)$$

$$F_2(x, y) = \frac{1}{3} \begin{bmatrix} x \\ y \end{bmatrix} + \frac{1}{3} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad (40)$$

$$F_3(x, y) = \frac{1}{3} \begin{bmatrix} x \\ y \end{bmatrix} + \frac{1}{3} \begin{bmatrix} 2 \\ 0 \end{bmatrix} \quad (41)$$

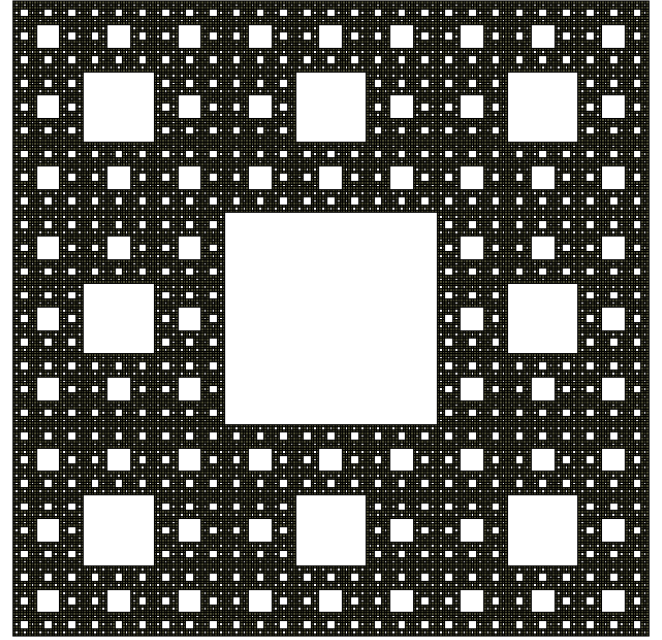
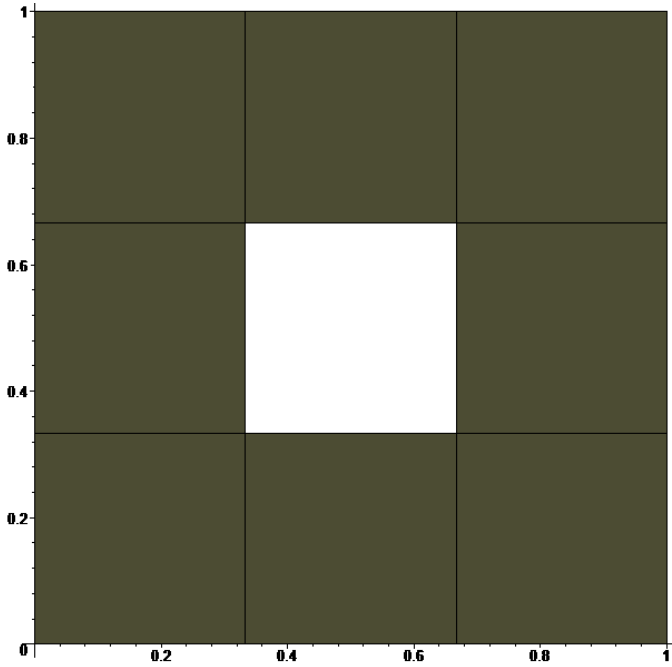
$$F_4(x, y) = \frac{1}{3} \begin{bmatrix} x \\ y \end{bmatrix} + \frac{1}{3} \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad (42)$$

$$F_5(x, y) = \frac{1}{3} \begin{bmatrix} x \\ y \end{bmatrix} + \frac{1}{3} \begin{bmatrix} 0 \\ 2 \end{bmatrix} \quad (43)$$

$$F_6(x, y) = \frac{1}{3} \begin{bmatrix} x \\ y \end{bmatrix} + \frac{1}{3} \begin{bmatrix} 1 \\ 2 \end{bmatrix} \quad (44)$$

$$F_7(x, y) = \frac{1}{3} \begin{bmatrix} x \\ y \end{bmatrix} + \frac{1}{3} \begin{bmatrix} 2 \\ 1 \end{bmatrix} \quad (45)$$

$$F_8(x, y) = \frac{1}{3} \begin{bmatrix} x \\ y \end{bmatrix} + \frac{1}{3} \begin{bmatrix} 2 \\ 2 \end{bmatrix} \quad (46)$$



Figure(5.7): Construction of the Sierpinski's Carpet. Figure(5.8): Sierpinski's Carpet.

The Sierpinski's Carpet is exactly self-similar to an every of its parts. Hence it is a fractal. To calculate the dimension of such an object we need 8 squares ($N_E = 8$) edge size $E = 1/3$, so by the Equation(7) we get:

$$D = \lim_{E \rightarrow 0} \frac{\log N_E}{-\log E} = \frac{\log 8}{-\log \frac{1}{3}} = \frac{\log 8}{\log 3} \simeq 1.89 \quad (47)$$

Hence the dimension of this figure is: $D \simeq 1.89$, which is bigger than the dimension of the Sierpinski's Sieve ($D \simeq 1.58$). Therefore we can assume that the Sierpinski's Carpet 'takes more space' than the Sierpinski's Triangle. The dimension of this figure is between a line and a plane, but 'very close' to the dimension of a plane.

We can notice that the relation between a number of similarity transformations ($m = 8$) and a number of elements ($N_E = 8$), they are equal. Therefore by estimation of the dimension we know how many Iterated Function Systems we need to construct a fractal. In the next chapter we will use these functions to produce fractal pictures.

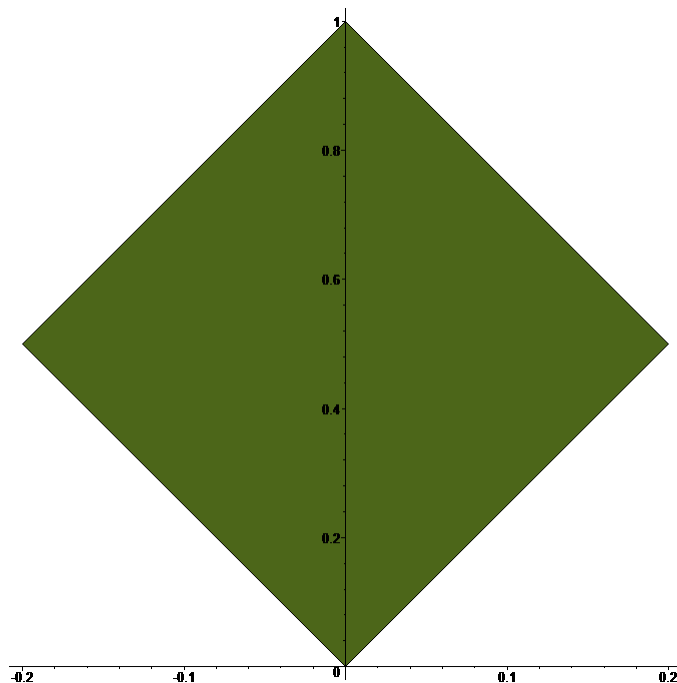
6 Implementation of Fractals in Maple.

To implement Iterated Function Systems numerically we develop the procedure to display fractal structures. We use mathematical and analytical software: Maple.

6.1 Elements of programming for IFS.

To display objects in Maple we can use polygons. The polygon is build by connecting points in the (x-y)-plane. For example if we want to display a rhombus we need four points, the Maple command is:

```
> PLOT(POLYGONS([[0,0], [-0.2,0.5], [0,1], [0.2,0.5],  
[0,0]]),COLOR(RGB,0.3,0.4,0.1));
```



Figure(6.1): Polygon plot of the rhombus.

Hence we need a list containing four elements, where each element describes location of a point: $L = [[0,0], [-0.2,0.5], [0,1], [0.2,0.5], [0,0]]$. Let call this list the initial list.

Next we want to apply m similarity transformations to our initial list. To implement the function which operates on lists we need following Maple command:

```
> fm:=x->[a,b]+c*[x[1],x[2]];
```

Where a, b, c , are constants used in the Equation(24).

Next we can use mapping command which applies functions to each element of the list.

The Maple command is:

```
> L1:=map(f1,L), map(f2,L), map(f3,L),map(f4,L), ..., map(fm,L);
```

So we can construct list of mappings. Hence this is a first iteration of the initial list. Let call it the motif of the fractal shape.

To get next iteration we can construct sequence of mappings. The Maple command is:

```
> S:=seq(map(f1,L1[i]),i=1..m^n), seq(map(f2,L1[i]),i=1..m^n),  
..., seq(map(fm,L1[i]),i=1..m^n);
```

Next, by putting this sequence in to a loop we can get required number of iterations.

Finally we convert our sequence into a list and display it in the polygon plot.

6.2 Maple Commands.

To implement fractals in the Maple software we need commands to display pictures and functions to plot them.

Plot commands:

- ▶ The 'PLOT' command produces the image which we want to display.
- ▶ The 'POLYGONS' command construct the polygon from a list. We can plot many polygons by putting them into one list.
- ▶ Plot options: we can specify the line thickness by: 'THICKNESS(n)' and colour by command: 'COLOR(RGB, 1.0,1.0,1.0)', where expression in the bracket denotes a hue.

Functions and lists:

- ▶ A list is an ordered sequence of expressions separated by comas and enclosed in square brackets '[...]'.
- ▶ To implement the function 'f' in Maple we assign it by: 'f:=x->f(x)', so we map x to f(x).
- ▶ The 'map' command applies the procedure to each operand of an expression. In our case applies the function to each element of the list.
- ▶ The 'seq' command creates a sequence. For example 'seq(f(i), i=1..n)' generates sequence: f(1), f(2), f(3), ..., f(n).
- ▶ To use a 'display' command we need to load the 'plots' package and we can display our fractal images.

6.3 Structure of the Code.

To make the programming easier and clearer to read and understand we structure our code and produce flow chart.

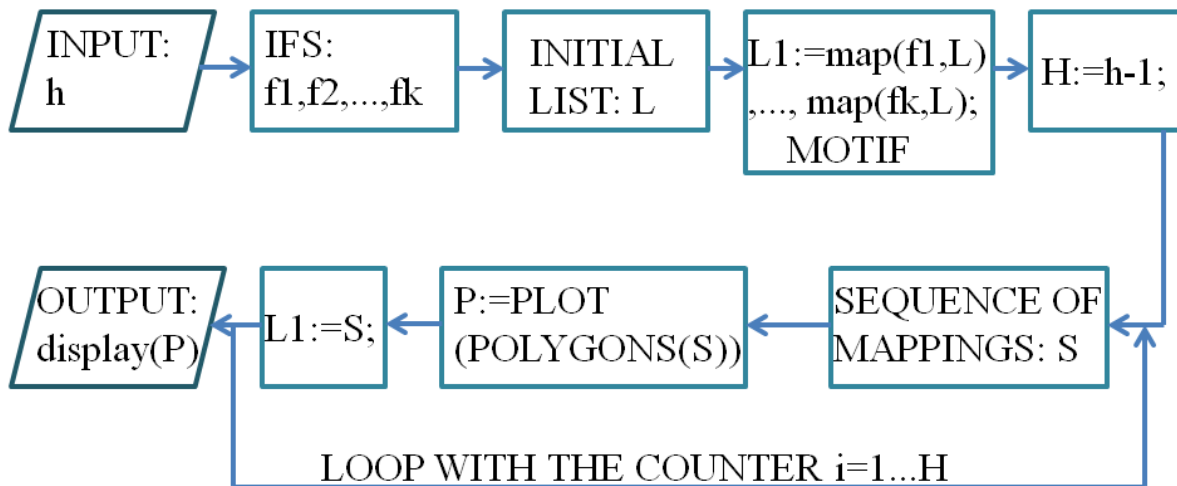
Structure of the code:

- input: number of iterations h
- implement Iterated Functions System
- assign L to the initial list
- assign L1 to first iteration to create a motif of the fractal shape
- transcript an input to start the loop from the second iteration
- assign S to the sequence of mappings
- plot polygons [S]
- transcript L1 to S
- output: displays the plot of polygons

The above structure is the base to all codes to produce fractal pictures in this essay. All these codes can be found in the Appendix.⁶

⁶ Codes in the Appendix contain comments to make them easier to read and understand.

Flow chart:



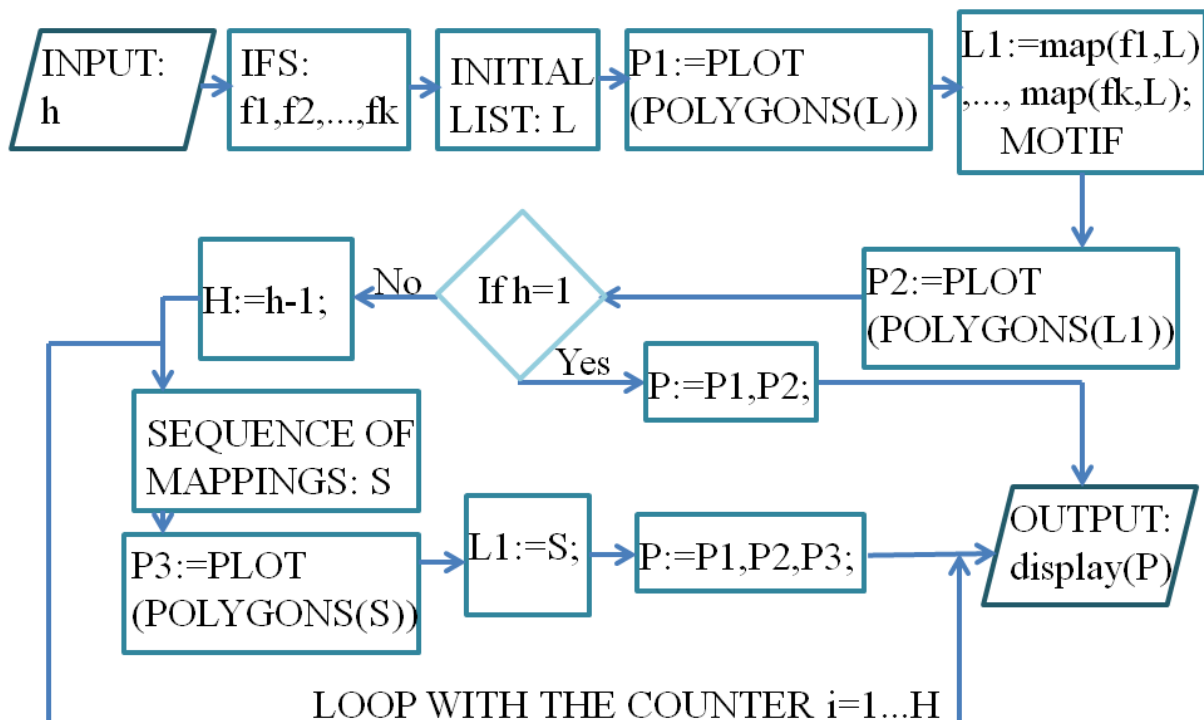
Figure(6.2): Flow chart to generate the Maple code to display the fractal structure.

The last detail we need to discuss is the sequence of mappings S . Let assume that we want to apply k contraction mappings. So S will contain k mappings. Each mapping command applies function to i -th element of the list, where i is the counter in the map command. To match the number of mappings with the number of elements in a list we need k^n operations. Where n denotes counter in the loop.

6.4 Pictures which Display Iterations Steps of the Fractal.

We can display all iterations plots in one picture, as the Cantor Set in the Figure(3.8). We need to add if statement and combine all polygon plots into one list, see Appendix(1).

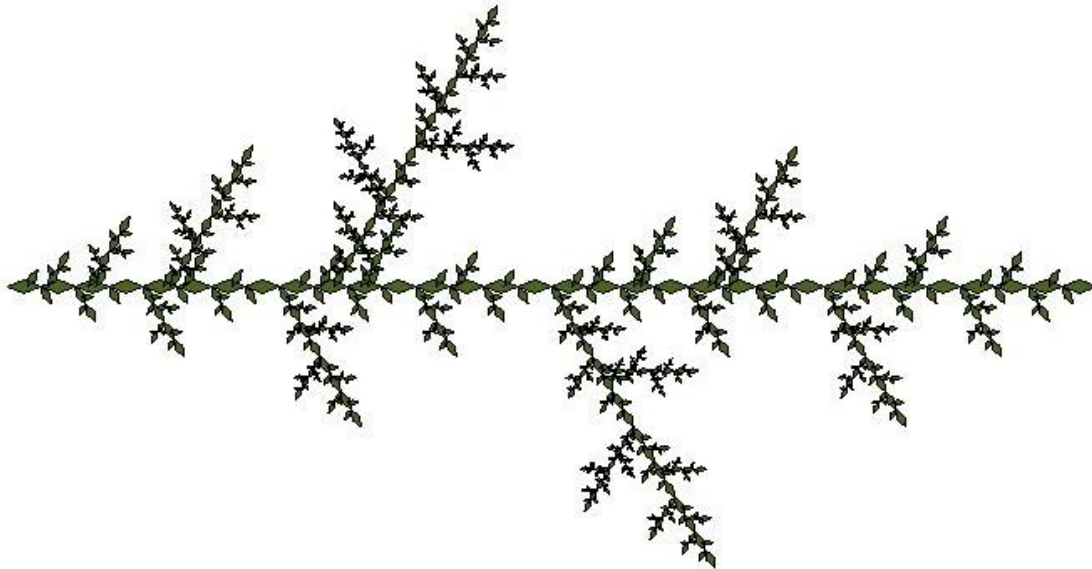
This is shown in the next flow chart:



Figure(6.3): Flow chart to the Maple code to display the fractal iterations on one picture.

6.5 Creating Own Fractal Pictures.

Now we have tool, the procedure to construct fractals. We can use our creativity to produce fractal images.



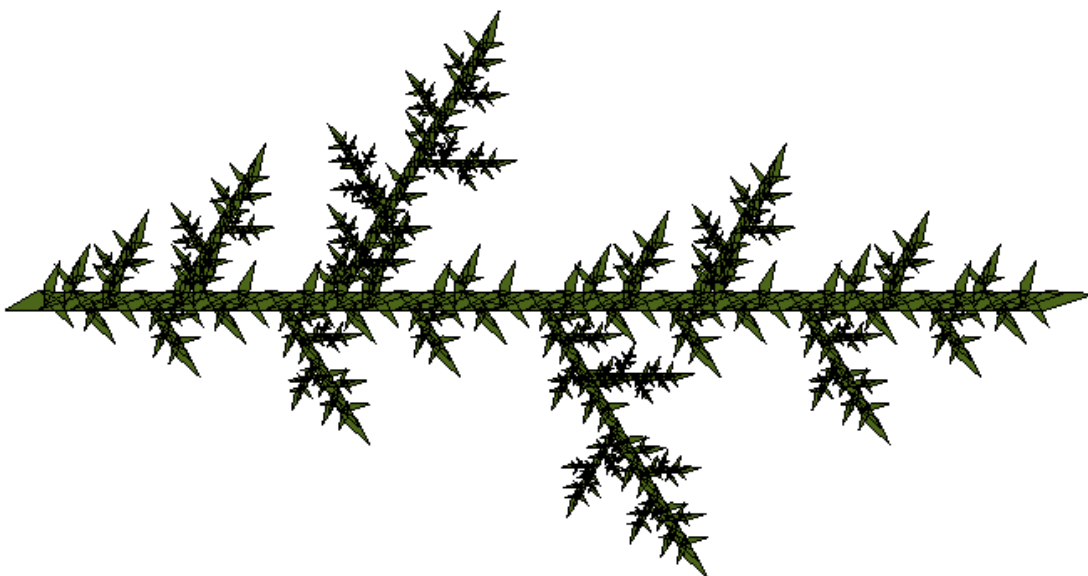
Figure(6.4): Fifth iteration of the Fractal Plant 1.

The above fractal image, Figure(6.4) is constructed by use of four contraction mappings, see Appendix(8). The dimension of this object by the Equation(7) is:

$$D = \lim_{E \rightarrow 0} -\frac{\log N_E}{\log E} = -\frac{\log 4}{\log_5 2} \approx 1.5123 \quad (48)$$

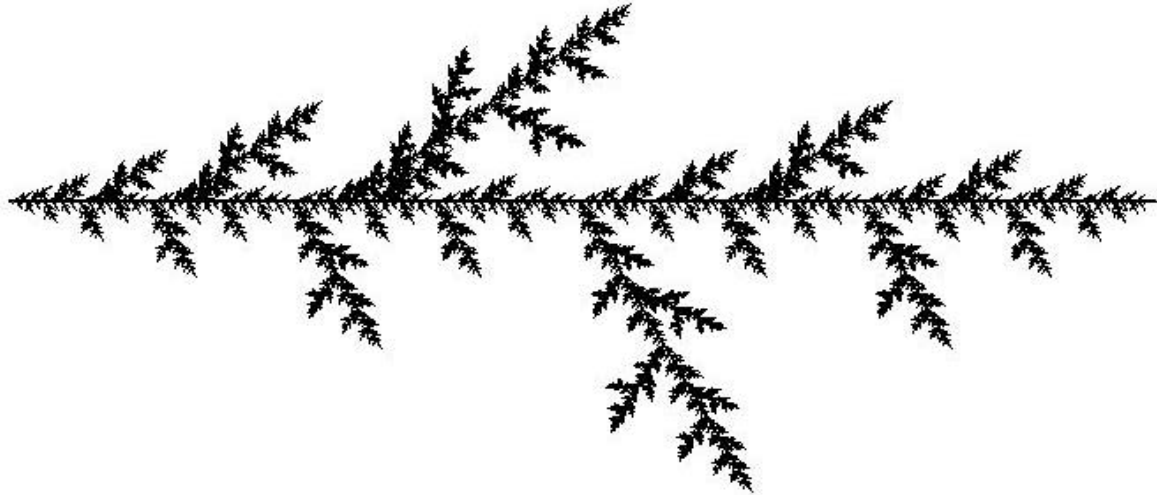
Next depending on our taste and imagination we can create more complicated pictures. Hence mathematical tools can give an artistic experience. As an example, in the next picture, Figure(6.5) we can see the same fractal as in the Figure(6.4) with small change to the shape of initial rhombus. We can perform this by changing the values in the initial list to:

$L = [[0.5, -0.5], [-0.1, 0.5], [0, 2], [0.5, 0.5]]$.



Figure(6.5): Fifth iteration of the modified Fractal Plant 1.

To get more realistic pictures we perform more iteration steps. This is shown on the next picture:



Figure(6.6): Seventh iteration of the modified Fractal Plant 1.

Next by modifying functions we can move objects in the (x-y)-plane. We can perform that by changing the value of the contraction factor, position vectors and rotation angles, Equation(23). The result of such modifications is shown in the next picture:



Figure(6.7): Seventh and ninth iteration of the Fractal Plant 2.⁷

The artificial picture in the Figure(6.7) is made up by three affine transformations, see Appendix(9). Hence we need only three functions to construct a natural object which looks like plant. These objects are exactly self-similar because they are made in recursive process of iterations. Therefore we can conclude that natural shapes can be described by fractal geometry, where an identical motif repeats itself on the decreasing scale.

⁷ All pictures in this chapter are my own ideas and I had a lot of joy doing them.

7 Summary.

In this Essay we have seen how to display exactly self-similar objects, fractals. Using the Box-Counting method we estimate their dimension, which do not need to be an integer. By application of similarity transformations to a set we construct an invariant set. We have learned how to display these Iteration Function Systems using programming tools in Maple. By changing these functions and an initial list in the code we are able to produce various fractal pictures, see an Appendix. To make the code more flexible we can modify it by implementing an initial list and contraction mappings as an input.

We developed the produce to produce our own fractals which can look like natural objects, for example 'plant' in the Figure(6.7). These objects are perfectly self-similar, as the Koch Curve. We have seen in the Chapter(3) that this curve mathematically represent coastlines. But coastlines are statistically self-similar. To get more realistic fractals we deform their structure using probabilistic aspect. Hence to improve our procedure we can add chance mechanism and display randomly deformed fractals.

Composing artificial pictures gives an artistic experience which is used for instance in science fiction films. Another very useful possibility is to produce 3-Dimensional fractal pictures. Hence the next programming task would be to develop the procedure to display 3-D images.