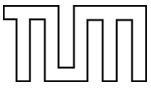

Technische Universität München

Department of Mathematics

Project Thesis

Fractals and their Simulation

Marvin Jahn

Advisor: Dr.rer.nat.habil. Heinz-Jürgen Flad

I assure the single handed composition of this project thesis only supported by declared resources. All the code is my own.

Munich, 05.05.2021 Marvin Jahn

Contents

1 Introduction

It is tempting to hope that the world around us can be modeled using simple shapes from Euclidean geometry like lines, circles, cubes etc. In many ways, Euclidean geometry can be seen as an "idealized" version of the world that is mathematically easier to work with. However, the vast majority of objects around us do not seem to permit a simple description in terms of Euclidean geometry but instead appear to be highly irregular. As Benoit B. Mandelbrot noted in [Man82]:

> "Clouds are not spheres, mountains are not cones, coastlines are not circles, and bark is not smooth, nor does lightning travel in a straight line."

Observations like these called for a new form of geometry, which would study objects of irregular shape. The resulting branch of mathematics is called *fractal geometry*.
As the name suggests, the central notion of this form of geometry is that of a *fractal*. Roughly, a fractal can be characterized as a geometric object, which when magnified reveals a similar structure as it had before. It will be our first task to make this notion of self-similarity precise. The main tool used for this are *iterated function systems*. In section 2.2, we state and proof *Hutchinson's Existence Theorem*, which constitutes a central pillar of our considerations. It not only guarantees the existence of many fractals, but it also supplies us with a rather simple algorithm that can be used to simulate any fractal given by an iterated function system. In the following subsection, we implement this algorithm in the programming language *Julia*. Section 2.4 can be regarded as a highlight of this paper: There we put our algorithm to work and derive images of a variety of popular and visually pleasing fractals. The algorithm also makes it easy to create and plot new fractals, which we demonstrate by an example.
In the next chapter, we study fractals that are not self-similar in the sense of Chapter 2, but instead permit a form of *statistical self-similarity*. Our focus lies on *Brownian motion* (also called *Wiener process*) and its generalization *fractional Brownian motion*. We provide a way to simulate Brownian motion as a limit of a random walk and we also describe and implement an algorithm that allows us to simulate fractional Brownian motion.
This text provides a short tour through the fascinating world of fractals and it is my hope that the various visualizations will delight the reader just as much as they delighted me.

2 Fractals as self-similar or self-affine Sets

2.1 Definition of Fractals

Many well-behaved objects as studied in Euclidean geometry lose structure and thus become easier to understand the more one "zooms in". For example, when magnified, a smooth curve looks more and more like its tangent. Furthermore, a large part of research in modern geometry is dedicated to manifolds, objects which are precisely defined by the fact that they are locally similar (homeomorphic) to Euclidean space $\mathbb{R}^n$. In contrast, fractals exhibit a fine structure, such that magnifying them reveals the same (or similar) structure.
In order to make this idea into a mathematical definition, one has to specify what is meant with "similar". There are many ways to make this precise and as fractal geometry attempts to describe a large class of phenomena appearing in reality, no single all-encompassing definition can be given.
For the purpose of specifying the forms of similarities we want to allow, it is convenient to fix some terminology.

Definition 2.1. Let (X, d), (Y, d) be two metric spaces and $f : X \to Y$ a function.

(a) f is called **contraction**, if it is Libschitz continuous with Libschitz constant $L \in (0, 1)$; i.e. if
$$d(f(x), f(y)) \leq L \cdot d(x, y) \quad \forall\, x, y \in X.$$

(b) f is called **similarity transformation**, if there exists $L > 0$, such that
$$d(f(x), f(y)) = L \cdot d(x, y) \quad \forall\, x, y \in X.$$

(c) We now restrict to the case that $X, Y \subset \mathbb{R}^n$ are two subsets of $\mathbb{R}^n$, equipped with the usual Euclidean metric. f is called **affine transformation**, if it is of the form $f(x) = g(x) + b$ with g a linear transformation and b a constant.

It is clear that all of these classes of functions are continuous.
We follow the terminology from [Fal14, Chapter 9].

Definition 2.2. Let (X, d) be a metric space. An **iterated function system** is a finite family of contractions $\{f_i : 1 \leq i \leq m\}$, $f_i : X \to X$. The corresponding **Hutchinson operator** is the function
$$H \colon \mathcal{P}(X) \to \mathcal{P}(X),\ S \mapsto \bigcup_{i=1}^{m} f_i(S).$$

We are mostly interested in the case that $X \subset \mathbb{R}^n$ is equipped with the Euclidean metric and we will only define fractals as subsets of $\mathbb{R}^n$, because this is the most significant case in practice.

Definition 2.3. Let $A \subset X \subset \mathbb{R}^n$. The set A is **invariant** with respect to an iterated function system $\{f_i : 1 \leq i \leq m\}$, $f_i : X \to X$, if we have $H(A) = A$ for the Hutchinson operator H.

Definition 2.4. A set $A \subset \mathbb{R}^n$ is **self-similar**, if it is invariant under an iterated function system consisting of similarity transformations. It is called **self-affine**, if it is invariant under an iterated function system consisting of affine transformations.

As mentioned, fractal geometry attempts to describe a large variety of irregular shapes that appear in reality. Due to this aspired generality, an all-encompassing definition of fractals cannot be given. In this chapter, we characterize fractals by the following property.

Definition 2.5. We call a set $F \subset \mathbb{R}^n$ a **fractal**, if (but not only if) it is self-affine or self-similar.

2.2 Hutchinson's Existence Theorem

A natural question that arises is whether an iterated function system $\{f_i : 1 \leq i \leq m\}$ necessarily permits an invariant set and whether such an invariant set is unique. Perhaps surprisingly, this is indeed the case if the domain X of the f_i is closed. This is a key result from Hutchinson in [Hut81, 3.1].

Theorem 2.6 (**Hutchinson's Existence Theorem**)**.** Let $X \subset \mathbb{R}^n$ be closed and $\{f_i : 1 \leq i \leq m\}$, $f_i : X \to X$ an iterated function system with Hutchinson operator H. The system has a unique invariant set F, which is nonempty and compact.
Additionally, for any nonempty compact set $A \subset X$ with $H(A) \subset A$, it applies $F = \bigcap_{n=0}^{\infty} H^n(A)$.

Our next goal is to prove this theorem, which turns out to be rather difficult. We first recall the *Hausdorff metric*, originally introduced by Felix Hausdorff in 1914.

Definition 2.7. Let (X, d) be a metric space and $\mathcal{C} \subset \mathcal{P}(X)$ the set of nonempty compact subsets of X. The **Hausdorff metric** is a metric on $\mathcal{C}$, given by

$$d_H \colon \mathcal{C} \times \mathcal{C} \to \mathbb{R},\ (A, B) \mapsto \max\left\{\sup_{a \in A} d(a, B), \sup_{b \in B} d(A, b)\right\}.$$

Intuitively, two nonempty compact sets are close to each other with respect to the Hausdorff metric, if any point in either of the two sets is close to another point from the other set.
The following lemma is an important ingredient for the proof of Theorem 2.6.

Lemma 2.8 ([Hut81, 3.2])**.** Let (X, d) be a metric space and $\{f_i : 1 \leq i \leq m\}$, $f_i : X \to X$ an iterated function system with Hutchinson operator H. Then the restriction $H : (\mathcal{C}, d_H) \to (\mathcal{C}, d_H)$ is a contraction.

Proof. Let $L_i \in (0, 1)$, $1 \leq i \leq m$ be the Libschitz constants of the f_i and $A, B \in \mathcal{C}$ nonempty compact sets.
For $a \in A$ and $1 \leq j \leq m$, we have for any $b \in B$

$$d\left(f_j(a), \bigcup_{i=1}^{m} f_i(B)\right) = \inf_{i \in \{1,\dots,m\},\ x \in B} d(f_j(a), f_i(x)) \leq d(f_j(a), f_j(b)),$$

so

$$\sup_{a\in A} d\left(f_j(a), \bigcup_{i=1}^{m} f_i(B)\right) \leq \sup_{a\in A} \inf_{b\in B} d(f_j(a), f_j(b)) = \sup_{a\in A} d(f_j(a), f_j(B)).$$

By symmetry, we also have

$$\sup_{b\in B} d\left(\bigcup_{i=1}^{m} f_i(A), f_j(b)\right) \leq \sup_{b\in B} d(f_j(A), f_j(b)),$$

and together, this implies

$$d_H\left(\bigcup_{i=1}^{m} f_i(A), \bigcup_{i=1}^{m} f_i(B)\right) \leq \max_{1\leq i\leq m} d_H(f_i(A), f_i(B)).$$

Thus the claim follows by the calculation

$$\begin{aligned} d_H(H(A), H(B)) &= d_H\left(\bigcup_{i=1}^{m} f_i(A), \bigcup_{i=1}^{m} f_i(B)\right) \\ &\leq \max_{1\leq i\leq m} d_H(f_i(A), f_i(B)) \\ &\leq \left(\max_{1\leq i\leq m} L_i\right) \cdot d_H(A, B). \end{aligned}$$

□

The Hausdorff metric can even be defined on the set of all nonempty closed bounded subsets $\mathcal{B}$, in which case the following statement applies.

Lemma 2.9 ([Mun13, Cha. 45, Exercise 7])**.** Let (X, d) be a complete metric space and $\mathcal{B}$ the set of all nonempty closed bounded subsets of X. Then $(\mathcal{B}, d_H)$ is a complete metric space.

Proof. Let $(A_n)_{n\in\mathbb{N}}$ be a Cauchy sequence in $(\mathcal{B}, d_H)$. By choosing a subsequence, we may assume that $d_H(A_n, A_{n+1}) < \frac{1}{2^n}$ for all $n \in \mathbb{N}$. We define $L \subset X$ to be the set of limits of sequences $(x_n)_{n\in\mathbb{N}}$ with $x_n \in A_n$ and $d(x_n, x_{n+1}) < \frac{1}{2^n}$ for all $n \in \mathbb{N}$.
The existence of such sequences is guaranteed because $d_H(A_n, A_{n+1}) < \frac{1}{2^n}$ and the corresponding limit point must exist since (x_n) is a Cauchy sequence and (X, d) is complete.
Having established that L is nonempty, we prove that L is bounded. Indeed, for $x \in L$, there exists a sequence $(x_n)_{n\in\mathbb{N}}$ with $x_n \to x$ satisfying the properties stated above. The same holds for $y \in L$ and a sequence $y_n \to y$. Since $x_n \to x$ and $y_n \to y$, there exists $n \in \mathbb{N}$, such that $d(x_n, x) < 1$ and $d(y_n, y) < 1$. Denoting the diameter of A_n by $r < \infty$, this implies

$$\begin{aligned} d(x, y) &\leq d(x, x_n) + d(x_n, y_n) + d(y_n, y) \\ &< 1 + d(x_n, y_n) + 1 \\ &\leq r + 2, \end{aligned}$$

so L is bounded. If a set in a metric space is bounded, the same holds true for its closure, so we have $\overline{L} \in \mathcal{B}$ by the above.
We finish the proof by showing that $A_n \to \overline{L}$. To that end, let $\epsilon > 0$.
We have to show that there exists $N \in \mathbb{N}$, such that for all $z \in \overline{L}$ and $a \in A_n$

$$d(z, A_n) < \epsilon \ \wedge \ d(\overline{L}, a) < \epsilon \quad \forall\, n \geq N. \tag{$*$}$$

Because $z \in \overline{L}$, there is $x \in L$ with $d(z, x) < \frac{\epsilon}{2}$ and a sequence $(x_n)_{n\in\mathbb{N}}$ satisfying $x_n \to x$ and $x_n \in A_n$. Hence there is $M \in \mathbb{N}$, such that $d(x_n, x) < \frac{\epsilon}{2}$ for all $n \geq M$. We conclude

$$d(z, A_n) \leq d(z, x_n) \leq d(z, x) + d(x, x_n) < \epsilon \quad \forall\, n \geq M.$$

To prove the second part of $(*)$, we choose $M' \in \mathbb{N}$ large enough, so that $\frac{1}{2^{M'-1}} < \epsilon$. We claim that for any $n \geq M'$ and $a \in A_n$, we can find $x \in L$ satisfying $d(x, a) < \epsilon$.
To see this, we define a sequence $(x_n)_{n\in\mathbb{N}}$ as follows. Set $x_n = a$ and iteratively choose $x_{n+k} \in A_{n+k}$, such that $d(x_{n+k-1}, x_{n+k}) < \frac{1}{2^{n+k-1}}$ for all $k > 0$. Those x_{n+k} must exist, because $d_H(A_i, A_{i+1}) < \frac{1}{2^i}$ for all $i \in \mathbb{N}$. Similarly, we define $x_1, \ldots, x_{n-1}$, such that $x_i \in A_i$ and $d(x_i, x_{i+1}) < \frac{1}{2^i}$. Because (X, d) is complete, we can find $x \in X$ with $x_n \to x$ and by construction, $x \in L$. For $k \geq n$, we observe

$$d(a, x) = d(x_n, x) \leq \sum_{i=n}^{k} d(x_i, x_{i+1}) + d(x_{k+1}, x) < \sum_{i=n}^{k} \frac{1}{2^i} + d(x_{k+1}, x)$$

and taking the limit $k \to \infty$ yields

$$d(a, \overline{L}) \leq d(a, x) < \frac{1}{2^{n-1}} \leq \frac{1}{2^{M'-1}} < \epsilon \quad \forall\, n \geq M'.$$

Now $(*)$ follows by taking $N \coloneqq \max\{M, M'\}$.

□

Equipped with the two results, the proof of Theorem 2.6 is not too difficult.

Proof of Theorem 2.6. Let $X \subset \mathbb{R}^n$ be a closed subset and consider it as a metric space with the usual Euclidean metric. By Lemma 2.8, the restriction of the Hutchinson operator $H : \mathcal{C} \to \mathcal{C}$ is a contraction with respect to the Hausdorff metric. Because X is closed and $\mathbb{R}^n$ is complete, the same holds true for X and we also have $\mathcal{B} = \mathcal{C}$. Therefore, Lemma 2.9 implies that $(\mathcal{C}, d_H)$ is a complete metric space. Banach's fixed point theorem guarantees the existence of a unique fixed point $F \in \mathcal{C}$; that is, $H(F) = F$. Furthermore, the theorem yields that $\lim_{n\to\infty} d_H(H^n(A), F) = 0$ for any $A \in \mathcal{C}$. If we additionally have $H(A) \subset A$, then $H^n(A)$ is a decreasing sequence, so by uniqueness of the limit we conclude $F = \bigcap_{n=0}^{\infty} H^n(A)$. □

It is interesting to note that Hutchinson's version of Theorem 2.6 ([Hut81, 3.1]) is more general, since he allows for arbitrary complete metric spaces. However, in that case special care has to be taken since in general $\mathcal{C} \subsetneq \mathcal{B}$.

The significance of this theorem is two-fold: On the one hand, it is of large theoretical importance as it guarantees the existence of many fractals. On the other hand, it allows us to approximate any fractal that is given by an iterated function system. The details of its usage for simulating fractals will be given in the following section.

2.3 Simulation of Fractals

Theorem 2.6 gives rise to the following algorithm, which can be used to approximate any set that is invariant under some iterated function system.

Algorithm 2.10 (**Simulation of Fractals**). Let $\{f_i : 1 \leq i \leq m\}$, $f_i : \mathbb{R}^n \to \mathbb{R}^n$ be an iterated function system with Hutchinson operator H. The corresponding invariant set F can be approximated as follows:

1. Choose a compact subset $A \subset \mathbb{R}^n$, such that $H(A) \subset A$.

2. For $n \in \mathbb{N}$, plot $H^n(A)$.
   Because $H(A) \subset A$, we have $H^{n+1}(A) \subset H^n(A)$ and thus the plotted set becomes smaller as n increases. By Theorem 2.6, $H^n(A)$ provides a good approximation of the invariant set F when n is large; and this approximation becomes exact in the limit.

Our next goal is to implement this algorithm in the programming language *Julia*, so that we can generate many interesting fractals. We restrict ourselves to the case $n = 2$ and assume that all f_i are affine transformations.
Even though the algorithm in written form is rather simple, its implementation is more involved. Because the algorithm essentially boils down to applying the Hutchinson operator to compact subsets of $\mathbb{R}^2$, it is first required to find a suitable way to represent such subsets of the plane in Julia. Luckily, it suffices for our purposes to work with polygons like rectangles or triangles instead of arbitrary compact sets. We may represent a polygon as an ordered vector (list) of vertices, where each vertex is given by a tuple representing its coordinates. For example, the unit square with lower left corner at the origin is given by

`[(0,0),(1,0),(0,1),(1,1)]`.

As a second step, we implement a function corresponding to the Hutchinson operator. However, instead of acting on arbitrary subsets of $\mathbb{R}^2$, our function is only defined on the set of polygons in $\mathbb{R}^2$. Because affine transformations preserve lines and parallelism, they also preserve polygons. The action of an affine transformation on a polygon is thus given by transforming each vertex individually. In Julia, our polygon-restricted version of the Hutchinson operator looks as follows:

```
function Hutchinson_operator(functions)
    return polygon -> [[f(x...) for x in polygon] for f in functions]
end
```

This function - called `Hutchinson_operator` - requires a list of functions `functions` as an argument and returns a function resembling the Hutchinson operator. The returned function takes a single polygon (here called `polygon`) as a parameter, and returns a list of polygons consisting of the images of that polygon under each of the functions from the supplied list. If `functions` is comprised of n functions, then the function returned by `Hutchinson_operator` calculates a list containing n polygons.

The crux of the algorithm is the iteration of the Hutchinson operator. If the f_i are fixed, then the function `Hutchinson_operator` returns a function g, which is our version of the Hutchinson operator. However, g maps a given polygon to a list of polygons, so it cannot be directly iterated. Instead, we want to apply g to every polygon in a given list, producing a list of lists of polygons, which will then be "flattened"; i.e. the lists will be concatenated to a single larger list. For example, when the supplied list consists of two polygons p_1, p_2, this can be visualized as follows:

$$[p_1, p_2] \xrightarrow{g} [[x_1, \ldots, x_n], [y_1, \ldots, y_n]] \xrightarrow{\text{concat}} [x_1, \ldots, x_n, y_1, \ldots, y_n].$$

In fact, this works much more generally: Let f be a function mapping instances of a data type A (in mathematical terms: elements of a set A) to a list of instances of data type B (in mathematical terms: an element of the free monoid of B). Then we can define an action of f on a list of instances of A just like we did in the case above, where A and B were the set of polygons: Apply f to each element in the supplied list and concatenate the results.
Readers familiar with category theory will realize that we just described the monad structure of the *list monad* (in mathematical terms: *free monoid monad*).
Unlike some purely functional programming languages (e.g. *Haskell*), *Julia* does not implement any instances of monads by default, but we can easily write the function we need ourselves:

```
concat(x) = vcat(x...)
bind(x, f) = concat(map(f,x))
```

Here `concat` is just the operation of concatenating ("flattening") a list of lists and `bind` corresponds to the action of a function `f` on a list `x` defined as above.

With these ideas in mind, the implementation of a function that iterates the Hutchinson operator is not hard.

```
function generate_fractal(functions,polygon,n)
    result = [polygon]
    for i in 1:n
        result = bind(result,Hutchinson_operator(functions))
    end
    return result
end
```

The function `generate_fractal` requires a list of functions (corresponding to the f_i in Algorithm 2.10), a "starting polygon" `polygon` (the set A in the algorithm) and the number of desired iterations n and returns the list of polygons generated by n-times iteration of the Hutchinson operator. This works by starting with a list consisting only of `polygon` (the unit of the list monad applied to `polygon`) and then iterating the Hutchinson operator using our defined `bind` function and a `for` loop.

Having defined `generate_fractal` such that it calculates the desired approximation as a list of polygons, we finish our implementation by writing a function to plot its result. For this, we import the *Plots* package:

```
using Plots
```

As the name suggests, this package can be used to plot many mathematical objects. In particular, it provides the `Shape` data type, which represents polygons in the same way as we did before. The advantage of this data type is that it can be easily plotted as the corresponding polygon. Equipped with this data type, we can finally plot our fractals:

```
1  function plot_fractal(functions,polygon,n)
2      xs = map(first,polygon)
3      ys = map(last,polygon)
4      xlim = (minimum(xs),maximum(xs))
5      ylim = (minimum(ys),maximum(ys))
6      if xlim[1] == xlim[2]
7          xlim = (xlim[1]-1,xlim[1]+1)
8      end
9      if ylim[1] == ylim[2]
10         ylim = (ylim[1]-1,ylim[1]+1)
11     end
12     polygons = generate_fractal(functions,polygon,n)
13     shapes = map(Shape, polygons)
14     result = plot(xlims = xlim, ylims = ylim, aspect_ratio=:equal,
15                   legend = false, border=:none)
16     map(shape -> plot!(shape,color = "black"), shapes)
17     display(result)
18 end
```

This code block defines the function `plot_fractal`, which embodies Algorithm 2.10 and has the same arguments as the function defined before. While the definition is rather long, the vast majority of code exists only for "visual purposes". Namely, in the lines 2 to 5, we determine the intervals in x and y-direction (called `xlim` and `ylim`, respectively), which our plot should show. We do this by examining our "starting shape" and using its minimal and maximal x-component (respectively y-component) as the boundary of the interval.
The lines 6 to 11 are of little importance; they only deal with the special case that the starting shape is 1-dimensional such that the calculated interval boundaries agree, in which case the respective interval is defined manually.
In line 12, we employ `generate_fractals` to calculate the list of polygons, which we then transform to the data type `Shape` (line 13), so that they can be easily plotted.
In the lines 14 and 15, we set a variety of visual parameters for our plot. This includes setting the intervals in x and y-direction to the values determined before, forcing equal scaling in both directions and removing the legend and coordinate axes. For details regarding the keywords consult the documentation [Bre].
Finally, we plot all the polygons in the list (line 16) and display the result (line 17).

This concludes our implementation of Algorithm 2.10. The generality of our implementation is quite remarkable: It allows us to simulate arbitrary fractals in $\mathbb{R}^2$, when we are given its iterated function system and a "starting polygon", which contains its own image

under the Hutchinson operator.
The price for this generality is the terrible runtime of the algorithm: Given m functions, the corresponding Hutchinson operator returns a list of m polygons when applied to a single polygon. Therefore, the n-th approximation of our fractal consists of m^n polygons. Each of the f_i is applied in every iteration to every polygon, so in total any f_i is applied

$$\sum_{i=0}^{n-1} m^i = \frac{m^n - 1}{m - 1}$$

times to a polygon. Furthermore, our polygon consists of k vertices, so each f_i is actually applied

$$\frac{m^n - 1}{m - 1} \cdot k$$

times. In total, our algorithm requires

$$\frac{m^n - 1}{m - 1} \cdot k \cdot m$$

function applications of the f_i.
It is clear that this exponential relationship in n forbids the calculation of the fractal for large n. Therefore, other algorithms (oftentimes adapted to the specific fractal at hand) are required to calculate more precise approximations.
Regardless, our algorithm produces surprisingly good and visually appealing results even for small n, which we will explore in the next section.

2.4 Examples of Fractals

We present a gallery of beautiful fractals (or rather, their approximations), utilizing our implementation from the previous section.
To use our function, we only have to define the functions of the iterated function system of the fractal and choose a "starting polygon". We will usually start with the unit square with lower left corner at the origin, but there is nothing special about it; any polygon that contains its own image under the Hutchinson operator can be used. While the images obtained when changing the "starting polygon" will be slightly different, in the limit all images must agree, as this is a consequence of Theorem 2.6.
In our examples, we will first offer the Julia code and then show the resulting images.

Example 2.11. A well-known fractal is the *Koch curve*, originally introduced by Helge von Koch in 1904. It is the fractal defined by the iterated function system

$$f_1(x, y) = \begin{bmatrix} \frac{1}{3}x \\ \frac{1}{3}y \end{bmatrix},$$

$$f_2(x, y) = \begin{bmatrix} \cos(\frac{1}{3}\pi) & -\sin(\frac{1}{3}\pi) \\ \sin(\frac{1}{3}\pi) & \cos(\frac{1}{3}\pi) \end{bmatrix} \begin{bmatrix} \frac{1}{3}x \\ \frac{1}{3}y \end{bmatrix} + \begin{bmatrix} \frac{1}{3} \\ 0 \end{bmatrix},$$

$$f_3(x, y) = \begin{bmatrix} \cos(\frac{5}{3}\pi) & -\sin(\frac{5}{3}\pi) \\ \sin(\frac{5}{3}\pi) & \cos(\frac{5}{3}\pi) \end{bmatrix} \begin{bmatrix} \frac{1}{3}x \\ \frac{1}{3}y \end{bmatrix} + \begin{bmatrix} \frac{1}{2} \\ \frac{\sqrt{3}}{6} \end{bmatrix},$$

$$f_4(x,y) = \begin{bmatrix} \frac{1}{3}x \\ \frac{1}{3}y \end{bmatrix} + \begin{bmatrix} \frac{2}{3} \\ 0 \end{bmatrix}.$$

Thus we can plot the Koch curve with the following Julia code:

```
f_1(x,y) = (1/3 * x, 1/3 * y)
f_2(x,y) = (1/3 + cos(1/3 * pi) * 1/3 * x - sin(1/3 * pi) * 1/3 * y,
            sin(1/3 * pi) * 1/3 * x + cos(1/3 * pi) * 1/3 * y)
f_3(x,y) = (1/2 + cos(5/3 * pi) * 1/3 * x - sin(5/3 * pi) * 1/3 * y,
            sqrt(3)/6 + sin(5/3 * pi) * 1/3 * x + cos(5/3 * pi) * 1/3 * y)
f_4(x,y) = (2/3 + 1/3 * x, 1/3 * y)
plot_fractal([f_1,f_2,f_3,f_4],[(0,0),(1,0),(1,1),(0,1)],n)
```

Figure 1: The Koch curve from $n = 0$ (top left) to $n = 5$ (bottom right).

Example 2.12. As a second example, we study the *Sierpiński triangle*, which can be plotted using the following code:

```
f_1(x,y) = (1/2 * x, 1/2 * y)
f_2(x,y) = (1/2 + 1/2 * x, 1/2 * y)
f_3(x,y) = (1/4 + 1/2 * x, 1/2 + 1/2 * y)
plot_fractal([f_1,f_2,f_3],[(0,0),(1,0),(1,1),(0,1)],n)
```

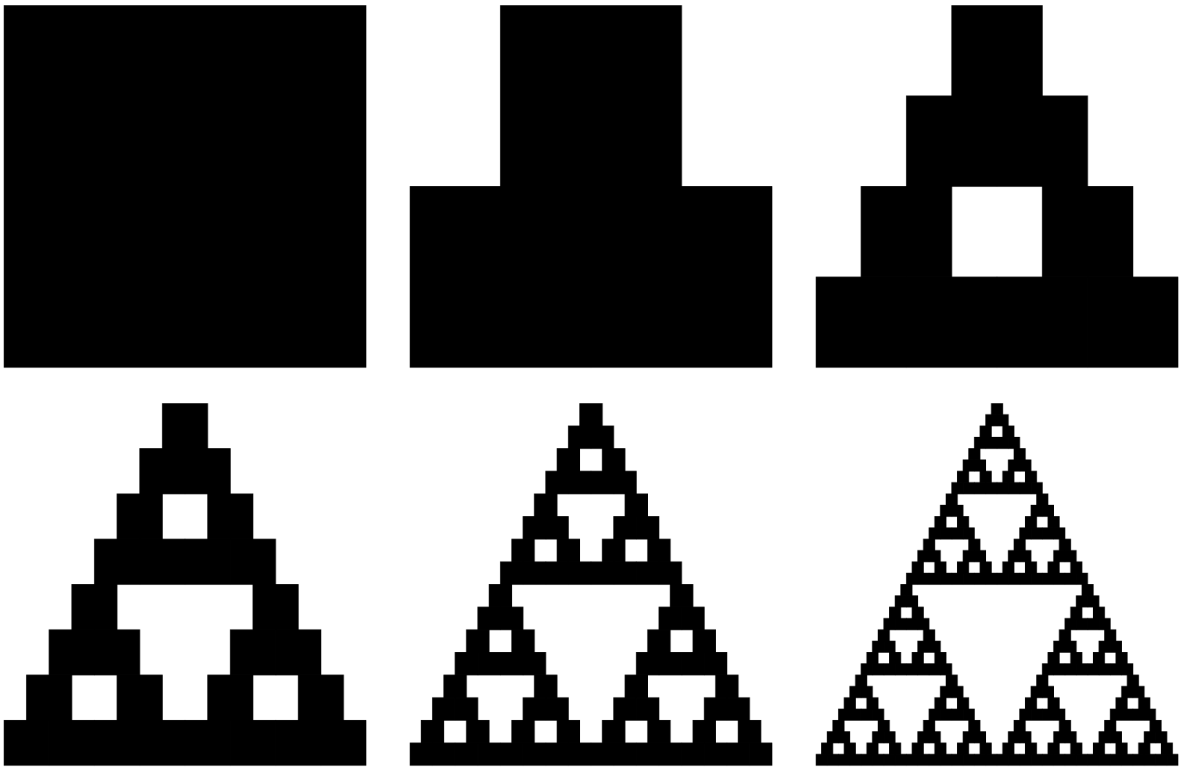

Figure 2: The Sierpiński triangle from $n = 0$ (top left) to $n = 5$ (bottom right).

Example 2.13. The *Sierpiński carpet* was originally defined by Sierpiński in 1916 and constitutes a generalization of the well-known Cantor set to two dimensions. We may plot it with the following code:

```
f_1(x,y) = (1/3 * x, 1/3 * y)
f_2(x,y) = (1/3 * x + 1/3, 1/3 * y)
f_3(x,y) = (1/3 * x + 2/3, 1/3 * y)
f_4(x,y) = (1/3 * x, 1/3 + 1/3 * y)
f_5(x,y) = (2/3 + 1/3 * x, 1/3 + 1/3 * y)
f_6(x,y) = (1/3 * x, 2/3 + 1/3 * y)
f_7(x,y) = (1/3 + 1/3 * x, 2/3 + 1/3 * y)
f_8(x,y) = (2/3 + 1/3 * x, 2/3 + 1/3 * y)
plot_fractal([f_1,f_2,f_3,f_4,f_5,f_6,f_7,f_8],
             [(0,0),(1,0),(1,1),(0,1)],n)
```

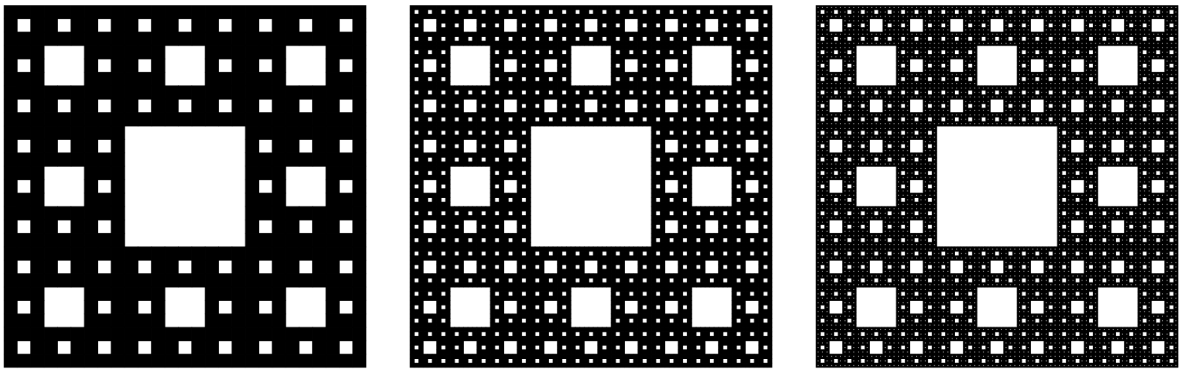

Figure 3: The Sierpiński carpet from $n = 3$ (left) to $n = 5$ (right).

Example 2.14. Another example is the *Vicsek fractal*, which we can plot as follows.

```
f_1(x,y) = (1/3 + 1/3 * x, 1/3 * y)
f_2(x,y) = (1/3 * x, 1/3 + 1/3 * y)
f_3(x,y) = (1/3 + 1/3 * x, 1/3 + 1/3 * y)
f_4(x,y) = (2/3 + 1/3 * x, 1/3 + 1/3 * y)
f_5(x,y) = (1/3 + 1/3 * x, 2/3 + 1/3 * y)
plot_fractal([f_1,f_2,f_3,f_4,f_5],[(0,0),(1,0),(1,1),(0,1)],n)
```

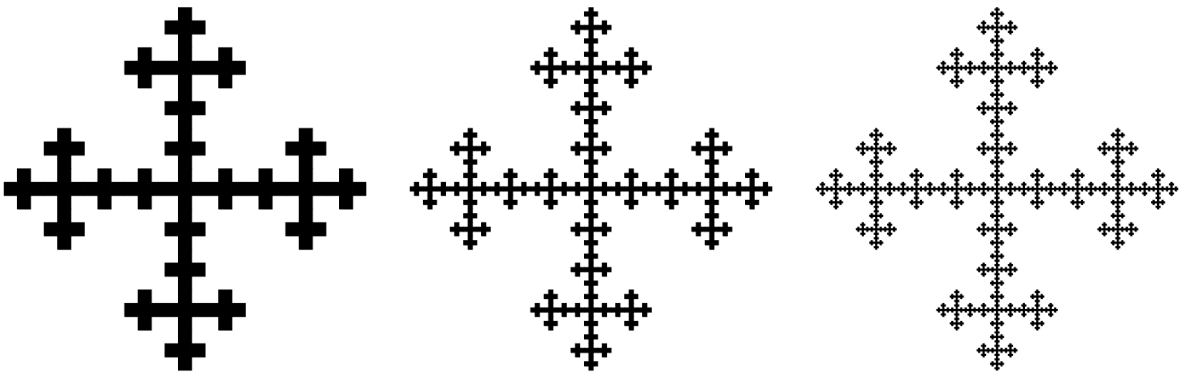

Figure 4: The Vicsek fractal from $n = 3$ (left) to $n = 5$ (right).

We have highlighted some well-known fractals and due to the large generality of our algorithm, the possibilities are almost endless. It is easy to change one of the defining functions to derive some other variant or to merely see which fractal structure reveals itself. But it is equally simple to define own, personal fractals. A final example shall illustrate this.

Example 2.15. The functions in the following code defines a fractal, which we name *butterfly fractal* (because parts of it look somewhat like a butterfly).

```
f_1(x,y) = (1/4 + 1/2 * x, 1/2 * y)
f_2(x,y) = (1/2 * x, 1/4 + 1/2 * y)
```

```
f_3(x,y) = (1/2 + 1/2 * x, 1/4 + 1/2 * y)
f_4(x,y) = (1/4 + 1/2 * x, 3/4 + 1/2 * y)
plot_fractal([f_1,f_2,f_3,f_4],[(0,0),(1,0),(1,1),(0,1)],n)
```

Figure 5: The butterfly fractal from $n = 3$ (left) to $n = 5$ (right).

We hope the presented examples convinced the reader that fractals are indeed beautiful. Even though the highlighted examples also looked rather artificial, fractals can also resemble naturally occurring objects. An example for this is the *Barnsley fern*, described by Michael Barnsley in his book [Bar93], which looks like a fern from nature. It is also an example of a fractal that employs randomization, which is the reason why it cannot be directly plotted with our algorithm.
In order to understand random fractals, we need to highlight another "form" of self-similiarity, called *statistical self-similarity*. While much can be said about *random fractals*, we choose a slightly different route and talk about statistical self-similarity in the context of *fractional Brownian motion*.

3 Fractals as statistically self-affine Sets

In the previous section, we have defined multiple notions describing how an object can be similar to itself at different scales. However, our definitions (see 2.4) are limited in the sense that they do not account for randomness. This is especially grave due to the fact that many "fractal-like" objects occurring in nature are not precise fractals but appear to have random deviations.
A useful tool to describe such objects is *fractional Brownian motion*, which constitutes a generalization of (ordinary) *Brownian motion*.

3.1 Brownian Motion

While Brownian motion for us is mostly a stepping stone towards fractional Brownian motion, its relevance in probability theory can hardly be understated. For example, Kallenberg calls it "arguably the single most important object of modern probability" [Kal21, p. 297].

In order to characterize *Brownian motion*, we need to define *random processes*. Their definition can be found in the standard literature of probability theory, e.g. [Kle20, Def. 9.1].

Definition 3.1. Let $(\Omega, \mathfrak{A}, \rho)$ be a probability space, $(X, \mathcal{L})$ a measurable space and I an index set. A **random process** (or **stochastic process**) is a collection $\{X(i) : i \in I\}$ of random variables $X(i) : \Omega \to X$.
Any $\omega \in \Omega$ gives rise to a **sample function**

$$X(-, \omega) \colon I \to X,\ i \mapsto X(i, \omega).$$

Here the space X is called the **state space**, because it consists of the different values that a random process may have.

We are mainly interested in the case that $(X, \mathcal{L}) = (\mathbb{R}, \mathfrak{B})$ are the real numbers $\mathbb{R}$, equipped with the Borel σ-algebra $\mathfrak{B}$ and $I = [0, \infty)$. Then the index set I can be interpreted as time. In that case, one can look at *increments* $X(t_2) - X(t_1)$ for $t_1, t_2 \in [0, \infty)$, $t_1 \leq t_2$ to estimate how the random process changes over a given time period.

Definition 3.2. A random process $\{X(i) : i \in I\}$ with $X(i) : \Omega \to \mathbb{R}$ is called **centered**, if we have $\mathbb{E}[X(i)] = 0$ for every $i \in I$.

A useful prerequisite for understanding Brownian motion are *random walks*, which will prove useful to us when simulating Brownian motion (Algorithm 3.6).

Example 3.3. An important and heavily studied random process is the *random walk*. For $i \in \mathbb{N} \setminus \{0\}$, let Y_i be a "coin flip"; i.e. a uniformly distributed random variable with

$$\mathbb{P}(Y_i = 1) = \mathbb{P}(Y_i = -1) = \frac{1}{2}$$

and assume that the Y_i are independent. Denote their n-th partial sum by $X_n \coloneqq \sum_{i=1}^{n} Y_i$. A **simple random walk** is a random process $\{X(i) : i \in \mathbb{N}\}$, where $X(0) = 0$ and $X(n) = X_n$ as above [Rev13, Chapter 1].
As a slight generalization, one can choose a *time interval* $\tau > 0$ and a *step size* $h > 0$ and take a step of size h at every multiple of τ. In the description above, we have $\tau = h = 1$. Then a random walk with time interval τ and step size h is a random process $\{X'(i) : i \in \tau\mathbb{N}\}$ with $X'(i) = h \cdot X(\frac{i}{\tau})$, where X is a simple random walk.
Clearly, we have

$$\mathbb{E}[X'(i)] = h \cdot \mathbb{E}\left[X\left(\frac{i}{\tau}\right)\right] = h \cdot \sum_{i=1}^{i/\tau} \mathbb{E}[Y_i] = 0,$$

so the random walk is centered.

The key difference between Brownian motion and other kinds of random processes lies in the fact that increments are assumed to be normally distributed and independent. This is made more precise in the following definition.

Definition 3.4. The **Brownian motion** (or **Wiener process**) is a random process $\{X(t) : t \in [0, \infty)\}$ with $X(t) : \Omega \to \mathbb{R}$, such that:

(a) It applies $X(0) = 0$ almost surely and $X : [0, \infty) \to \mathbb{R}$ is continuous almost surely.

(b) For any $t \geq 0$ and $h > 0$, the increment $X(t + h) - X(t)$ is normally distributed with mean 0 variance h.

(c) Pairs of increments are independent; i.e. for $0 \leq t_1 \leq t_2 \leq \cdots \leq t_{2m}$, the increments $X(t_2) - X(t_1)$, $X(t_4) - X(t_3)$, $\ldots$, $X(t_{2m}) - X(t_{2m-1})$ are independent.

For $t \in [0, \infty)$, (a) implies that $X(t) = X(t) - X(0)$ almost everywhere, so by (b), $X(t)$ is normally distributed with mean 0 and variance t. In particular, $X(t)$ is centered.
A priori, it is not clear whether a random process with the desired properties actually exists. That this is indeed the case can be proven by considering random walks with time intervals $\tau > 0$ and taking the limit $\tau \to 0$.

Theorem 3.5 ([Kuo06, Theorem 1.2.2])**.** Let $\{X_\tau(i) : i \in \tau\mathbb{N}\}$ be a random walk with time interval $\tau > 0$ and step size $h \coloneqq \sqrt{\tau}$. For every $t \in [0, \infty)$, the limit

$$B(t) \coloneqq \lim_{\tau \to 0} X_\tau(i)$$

exists in distribution and B constitutes Brownian motion.

Due to this theorem, we may interpret Brownian motion as the continuous limit of a collection of discrete random walks. It also gives rise to the following simple algorithm, which can be used to simulate Brownian motion.

Algorithm 3.6 (**Simulation of Brownian motion**)**.**

1. Choose a finite set of evenly spaced points $0 = x_0 < x_0 + \tau < x_0 + 2\tau < \cdots < x_0 + (m-1)\tau = x_1$.

2. Simulate a random walk X with time interval τ and step size $h \coloneqq \sqrt{\tau}$. By Theorem 3.5, this provides a good approximation of Brownian motion when τ is small.

We now implement the algorithm in Julia.

```
function random_walk(tau,x)
    points = range(0, x, step = tau)
    h = sqrt(tau)
    m = length(points)
    Y = map(b -> if b h else -h end, rand(Bool,m))
    values = [0.0]
    for i in 2:m
        new_value = values[i-1] + Y[i]
        push!(values, new_value)
    end
    return points, values
end
```

This defines the function `random_walk`, which simulates a random walk with time interval `tau` > 0, starting at 0 and ending at the time specified by the parameter `x`. First, we generate the set of points `points` for which we want to calculate the random walk (line 2) and set the step size `h` appropriately (line 3). In line 5, we draw the random values representing our steps (i.e. each value is `h` with probability $\frac{1}{2}$ and otherwise $-h$) and save them in the vector `Y`. Having initialized the first value to be 0 in line 6, we determine the value of our random walk at each time > 0 in `points` by iterating from $i = 2$ to $i = m$ (line 7).
The new value is then calculated by adding the i-th entry of `Y` (the value of the i-th "coin flip") to the previous value and the result is inserted into the list (vector) `values` (line 9). Finally, we return the points and their corresponding values of our random walk.

Plotting the linear interpolation of the produced values of the `random_walk` function yields graphs like the following:

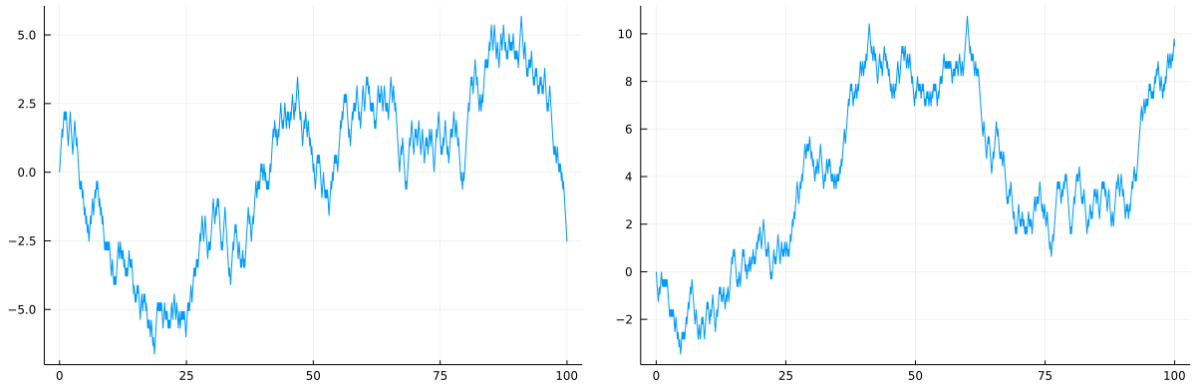

Figure 6: Two simulations of Brownian motion with `tau` $= 0.1$ and `x` $= 100$.

As mentioned, our interest in Brownian motion is mainly motivated by the desire to describe "random" fractals and we have yet to make this connection precise. It is apparent from Figure 6 that a sample graph of Brownian motion is endowed with some form of self-similarity, but the terminology from the previous chapter is not enough to capture this. Indeed, since Brownian motion is non-deterministic, its sample functions cannot be self-similar in the sense of Definition 2.4. Therefore, we adapt the notion of self-similarity in such a way that it applies to random processes.

Definition 3.7. A random process $\{X(t) : t \in [0, \infty)\}$ with $X(t) : \Omega \to \mathbb{R}$ is called **statistically self-affine**, if there exists $\alpha > 0$, such that for any $r, u > 0$, the random variables $X(ru)$ and $r^{\alpha}X(u)$ have the same distribution.

Intuitively, this means that changing the temporal scale by a factor r and the other scale by a factor $r^{-\alpha}$, we obtain a random process with the same probability distribution.
Using this new notion of self-similarity, we can modify Definition 2.5.

Definition 3.8. A set $F \subset \mathbb{R}^n$ is called a **fractal**, if (but not only if) it is self-affine, self-similar or the graph of a sample function of a statistically self-affine random process.

Proposition 3.9. The Brownian motion $\{X(t) : t \in [0, \infty)\}$ is statistically self-affine. In particular, its sample functions are fractals (in the sense of Definition 3.8).

Proof. Let $r, u > 0$. By (a), the distributions of $X(ru)$ and $r^{\frac{1}{2}}X(u)$ agree with those of $X(ru) - X(0)$ and $r^{\frac{1}{2}}(X(u) - X(0))$, respectively. Because scaling a normally-distributed random variable by a scalar $\lambda \in \mathbb{R}$ results in another normally-distributed random variable with the same mean and λ^2-scaled variance, (b) implies that both those increments are normally distributed with mean 0 and variance ru. □

3.2 Fractional Brownian Motion

It turns out that ordinary Brownian motion is quite restrictive in the sense that we cannot control how "rough" the resulting graphs are. This problem is solved by generalizing Brownian motion to fractional Brownian motion and introducing a parameter H (the *Hurst exponent*) that controls this "roughness".
The main feature of fractional Brownian motion lies in the fact that increments may depend on each other. This stands in direct contrast to ordinary Brownian motion, where pairs of increments are assumed to be independent. Intuitively, this means that the change of the process as time advances is dependent on how the process has changed before.
To put fractional Brownian motion in the proper theoretical context, we introduce an important class of random processes, called *Gaussian processes*.

Definition 3.10. A random process $\{X(t) : t \in [0, \infty)\}$ with $X(t) : \Omega \to \mathbb{R}$ is a **Gaussian process**, if for all $t_1, \ldots, t_m \in [0, \infty)$ and $\lambda_1, \ldots, \lambda_m \in \mathbb{R}$, the random variable $\sum_{i=1}^{m} \lambda_i X(t_i)$ is normally distributed.

Now we can define fractional Brownian motion.

Definition 3.11. The **fractional Brownian motion** with **Hurst exponent** $H \in (0, 1)$ is a Gaussian process $\{X(t) : t \in [0, \infty)\}$ such that:

(a) We have almost surely $X(0) = 0$ and $X : [0, \infty) \to \mathbb{R}$ is almost surely continuous.

(b) For any $t \geq 0$ and $h > 0$, the increment $X(t + h) - X(t)$ is normally distributed with mean 0 variance h^{2H}.

Just like for ordinary Brownian motion it follows directly from the definition that $X(t)$ is normally distributed with mean 0 and variance t^{2H}, implying that fractional Brownian motion is centered.
It is interesting to note that the original definition from Mandelbrot and Van Ness characterizes fractional Brownian motion by fixing a starting value $X_H(0)$ and then setting

$$X_H(t) \coloneqq X_H(0) + \frac{1}{\Gamma(H + \frac{1}{2})}\Bigg(\int_{-\infty}^{0} (t - s)^{H - \frac{1}{2}} - (-s)^{H - \frac{1}{2}} dB(s) + \int_{0}^{t} (t - s)^{H - \frac{1}{2}} dB(s) \Bigg).$$

Here Γ denotes the Gamma function and B corresponds to the *white noise measure* [MV68, Def. 2.1]. In fact, fractional Brownian motion had already been considered by

Kolmogorov in [Kol40].

There is an alternative characterization of fractional Brownian motion, which is common in the literature (see e.g. [Nou12, Def. 2.1]). We state it and prove that it is equivalent to our definition in the following theorem.

Theorem 3.12. The random process $\{X(t) : t \in [0,\infty)\}$ is fractional Brownian motion with Hurst exponent $H \in (0,1)$ if and only if it is a centered Gaussian process $\{X(t) : t \in [0,\infty)\}$ such that:

(a) The function $X : [0,\infty) \to \mathbb{R}$ is almost surely continuous.

(b) For all $s,t \in [0,\infty)$, X has the covariance function

$$\operatorname{Cov}(X(s),X(t)) = \frac{1}{2}\Big(s^{2H} + t^{2H} - |t-s|^{2H}\Big). \tag{$*$}$$

Proof. We have seen that fractional Brownian motion is centered, so in order to show one direction, we only have to check that the covariance function of a fractional Brownian motion is given by the formula $(*)$. For this we notice that if $s = t$, then

$$\operatorname{Cov}(X(s),X(s)) = \mathbb{E}(X(s)^2) = \operatorname{Var}(X(s)) = s^{2H} = \frac{1}{2}\big(s^{2H} + s^{2H}\big),$$

so the formula holds. If $s \neq t$, then because both the covariance and the formula $(*)$ are symmetric, we may assume that $s > t$. The assumption (b) in the definition of fractional Brownian motion implies

$$\mathbb{E}\big[(X(s) - X(t))^2\big] = \operatorname{Var}(X(s) - X(t)) = (s-t)^{2H},$$

so we conclude

$$\begin{aligned}
\operatorname{Cov}(X(s),X(t)) &= \mathbb{E}[X(s)X(t)] \\
&= \frac{1}{2}\cdot\mathbb{E}\big[-(X(s)-X(t))^2 + X(s)^2 + X(t)^2\big] \\
&= \frac{1}{2}\cdot\big(-\mathbb{E}\big[(X(s)-X(t))^2\big] + \mathbb{E}[X(s)^2] + \mathbb{E}[X(t)^2]\big) \\
&= \frac{1}{2}\cdot\big(-(s-t)^{2H} + s^{2H} + t^{2H}\big).
\end{aligned}$$

On the other hand, suppose $\{X(t) : t \in [0,\infty)\}$ is a centered Gaussian process satisfying (a) and (b). Since X is centered and

$$\operatorname{Var}(X(0)) = \operatorname{Cov}(X(0),X(0)) = 0,$$

it follows $X(0) = 0$ almost surely. Furthermore, as X is a centered Gaussian process, for $t \geq 0$ and $h > 0$, the increment $X(t+h) - X(t)$ is normally distributed with mean 0 and variance

$$\begin{aligned}
\operatorname{Var}(X(t+h) - X(t)) &= \mathbb{E}\big[(X(t+h) - X(t))^2\big] \\
&= \mathbb{E}\big[X(t+h)^2\big] - 2\mathbb{E}[X(t+h)X(t)] + \mathbb{E}\big[X(t)^2\big] \\
&= \operatorname{Var}(X(t+h)) - 2\operatorname{Cov}(X(t+h),X(t)) + \operatorname{Var}(X(t)) \\
&= (t+h)^{2H} - (t+h)^{2H} - t^{2H} + h^{2H} + t^{2H} = h^{2H}.
\end{aligned}$$

□

Just as in the case of ordinary Brownian motion, it must be proven that fractional Brownian motion actually exists. However, this proof turns out to be rather involved, requiring advanced results from probability theory. Therefore, we will omit the proof and instead redirect the interested reader to [Nou12, Prop. 1.6]. There it is shown that a random process with the properties described in Theorem 3.12 exists, so by that theorem the same holds for fractional Brownian motion as we have defined it.

Notice that for $H = \frac{1}{2}$, fractional Brownian motion just becomes ordinary Brownian motion. It is not surprising that they share the key property of statistical self-similarity.

Proposition 3.13. Fractional Brownian motion is statistically self-affine.

Proof. This follows with exactly the same proof as Proposition 3.9, replacing $\frac{1}{2}$ by H. □

We now want to present an algorithm which simulates fractional Brownian motion and implement it as a Julia function. The chosen algorithm and multiple alternatives are described in [Jea00].

Algorithm 3.14 (**Simulation of fractional Brownian motion with Hurst exponent** $H \in (0,1)$)**.**

1. Choose a finite set of points $0 < t_1 < \cdots < t_m$ for which the value of a sample of fractional Brownian motion X will be calculated.
2. Calculate the covariance matrix A for those points, which according to Theorem 3.12 is given by
$$A_{i,j} = \operatorname{Cov}(X(t_i), X(t_j)) = \frac{1}{2}\Big({t_i}^{2H} + {t_j}^{2H} - |t_j - t_i|^{2H}\Big).$$
3. Perform a Cholesky-decomposition of A; i.e. find a $m \times m$-matrix M with $M \cdot M^T = A$.
4. Let $V \in \mathbb{R}^m$ be a vector of independent standard normally distributed random variables. Then the i-th entry of the vector $X \coloneqq M \cdot V$ is the value of our sample at time t_i; that is, $X(t_i) \coloneqq X_i$ is the desired sample function.

The reason that we require $t_1 > 0$ is that otherwise the first column of A will be zero, implying that A is not an Hermitian positive-definite matrix, so the Cholesky decomposition wouldn't exist.
To see that the algorithm indeed yields a sample function of fractional Brownian motion with Hurst exponent $H \in (0,1)$, we check that X satisfies the equivalent characterization of Theorem 3.12 (or rather a discrete version of it, where $[0,\infty)$ is replaced by the set $\{t_i : 1 \le i \le m\}$). Indeed, we have

$$\mathbb{E}[X(t_i)] = \mathbb{E}\left[\sum_{j=1}^{m} M_{i,j} V_j\right] = \sum_{j=1}^{m} M_{i,j}\mathbb{E}[V_j] = 0$$

and the calculation

$$\mathbb{E}\big[X \cdot X^T\big] = \mathbb{E}\big[M \cdot V \cdot V^T \cdot M^T\big] = M \cdot \mathbb{E}\big[V \cdot V^T\big] \cdot M^T = M \cdot M^T = A,$$

implies that

$$\mathrm{Cov}(X(t_i), X(t_j)) = \mathbb{E}\Big[\big(X \cdot X^T\big)_{i,j}\Big] = A_{i,j}.$$

Because a linear combination of independent, normally distributed random variables is again normal, the process is Gaussian.

We now implement Algorithm 3.14 in the programming language *Julia*. First, we import two required packages, one for the simulation of probability distributions and the other for common operations of linear algebra.

```
using Distributions
using LinearAlgebra
```

The following code block defines the function `fractional_brownian_motion`, which will be used to simulate fractional Brownian motion.

```
1  function fractional_brownian_motion(H,x_0,x_1,tau)
2      points = range(x_0, x_1, step = tau)
3      m = length(points)
4      V = rand(Normal(),m)
5      A = zeros(m,m)
6      for i in 1:m
7          for j in 1:m
8              s = points[i]
9              t = points[j]
10             A[i,j] = 1/2 * (s^(2H) + t^(2H) - abs(s - t)^(2H))
11         end
12     end
13     M = cholesky(A)
14     return points, M.L * V
15 end
```

The function requires four arguments; the Hurst exponent `H` and three other parameters which are used in line 2 to determine the set of points for which a sample of fractional Brownian motion will be calculated. More precisely, we generate a set of uniformly spaced points, starting at `x_0` and ending at `x_1` with a step width of `tau`. In the lines 5-12, we calculate the covariance matrix A. This corresponds to step 2 of Algorithm 3.14. The Cholesky decomposition of step 3 is performed in line 13. Its result is multiplied with the vector `V`, which we generated in line 4 using standard normal distribution.

Using this function, we can plot fractional Brownian motion for various values of the Hurst exponent H. It is obvious from the graphs on the next page that a low value of H corresponds to a rougher graph, whereas large values of H correspond to a smoother one.

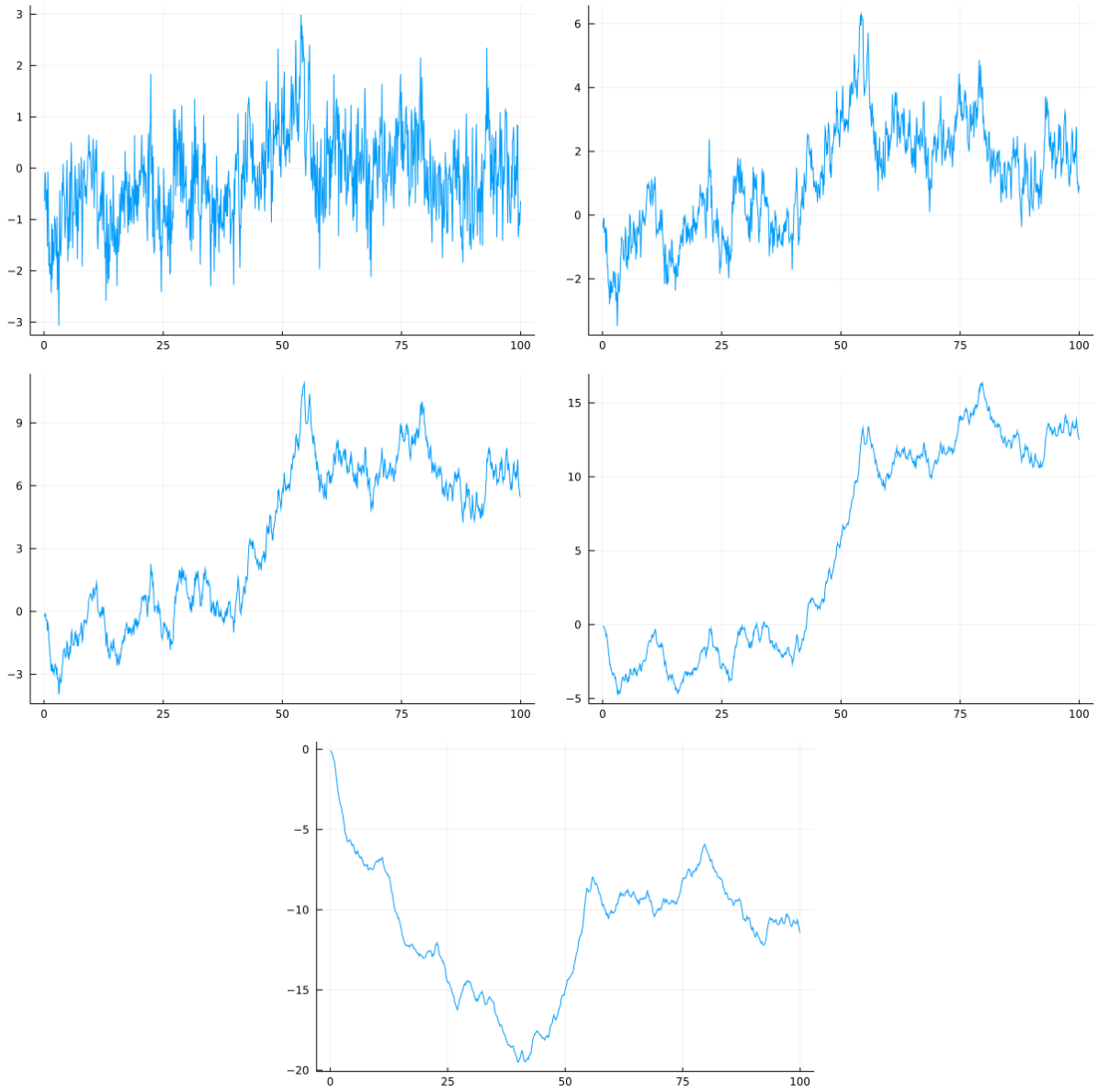

Figure 7: Simulations of fractional Brownian motion with `x_0` = 0, `x_1` = 100, `tau` = 0.1 and H increasing from 0.1 (top left) to 0.9 (bottom) by steps of 0.2.

4 Conclusion

Due to the enormous size of fractal geometry we could only give a brief overview of its key notion - the fractal. The next natural step would be to search for parameters that describe them. Inspired by Euclidean geometry, the notion of dimension is an obvious candidate. Indeed, dimension theory has established itself as a central pillar in the study of fractals and its applications. However, conventional notions of dimension like the *topological dimension* turn out to be non-sufficient, since they ignore the fine structure of fractals. Therefore, a large number of alternative definitions of a *fractal dimension* have been introduced, like the *Hausdorff dimension*, the *box-counting dimension* or the *simi-*

larity dimension [MF03].
As mentioned, fractal geometry was inspired and motivated by many phenomena occurring in the real world. Thus, it is not surprising that they prove useful in physics and many related fields.
However, it is important to realize that the fractals encountered in nature are not self-similar under arbitrary magnifications, but only for a certain range of length scales. Regardless, a large number of phenomena occurring in reality have been convincingly modeled using the tools of fractal geometry. For example, many experiments have shown that cracks can be modeled using fractals (see e.g. [MPP84] or [Hin+08]). There are also attempts to model cracks using fractional Brownian motion, see [Add+12].
Using the fractal dimension, it is possible to characterize fractals and distinguish them from scratches, which are harmless in the context of many applications [SVR95]. Brownian motion inherently has many physical applications, as it constitutes the mathematical model of the physical movement of particles, originally observed by Robert Brown in 1827 [Fal14, p. 279].
While applications of fractals in physics are probably the most prominent, they also reveal themselves in many other areas like geology (see e.g. [Car97]). A particularly well-known example of this is Mandelbrot's calculation of the fractal dimension of the coast of Great Britain in [Man67]. Fractals are also employed in the study of finances to model the behavior of prices of assets in stock markets [Nua06].
Another somewhat surprising application of fractals and iterated function systems (see Definition 2.2) appears in *fractal image compression*. The main goal of this area of computer science is to store images on a computer in such a way that the amount of occupied memory is as small as possible [Fis95].
The highlighted applications show that fractals are not only visually intriguing but also serve a "higher purpose" for many disciplines of science. The simulation of fractals thus constitutes an important tool for creating, verifying and improving models in those areas of science.

References

[Add+12] P.S. Addison et al. "A Fractional Brownian Motion Model of Cracking". In: (Nov. 2012), p. 7.

[Bar93] Michael F. Barnsley. *Fractals Everywhere*. Boston: MA: Academic Press, June 1993. ISBN: 978-0-12-079061-6.

[Bre] Thomas Breloff. *Julia Plots Homepage*. https://docs.juliaplots.org/latest/.

[Car97] James R. Carr. "Statistical Self-Affinity, Fractal Dimension, and Geologic Interpretation". In: *Engineering Geology*. Fractals in Engineering Geology 48.3 (Dec. 1997), pp. 269–282. ISSN: 0013-7952.

[Fal14] Kenneth Falconer. *Fractal Geometry: Mathematical Foundations and Applications*. 3rd edition. Hoboken: Wiley, Feb. 2014. ISBN: 978-1-119-94239-9.

[Fis95] Yuval Fisher. *Fractal Image Compression: Theory and Application*. New York: Springer-Verlag, 1995. ISBN: 978-1-4612-7552-7.

[Hin+08] Moisés Hinojosa et al. "Scaling Properties of Slow Fracture in Glass: From Deterministic to Irregular Topography". In: *International Journal of Fracture* 151.1 (Aug. 2008), p. 81. ISSN: 1573-2673.

[Hut81] John E. Hutchinson. "Fractals and Self Similarity". In: *Indiana University Mathematics Journal* 30.5 (1981), pp. 713–747.

[Jea00] Coeurjolly Jean-Francois. "Simulation and Identification of the Franctional Brownian Motion: A Bibliographical and Comparative Study". In: *Journal of statistical software* 5 (2000), pp. 1–53.

[Kal21] Olav Kallenberg. *Foundations of Modern Probability*. Third. Probability Theory and Stochastic Modelling. Springer International Publishing, 2021. ISBN: 978-3-030-61870-4.

[Kle20] Achim Klenke. *Probability Theory: A Comprehensive Course*. Third. Universitext. Springer International Publishing, 2020. ISBN: 978-3-030-56401-8.

[Kol40] Andrei N. Kolmogorov. "Wienersche Spiralen Und Einige Andere Interessante Kurven Im Hilbertschen Raum". In: *Comptes Rendus (Doklady) de l'Académie des Sciences de l'URSS* 26 (1940), pp. 115–118.

[Kuo06] Hui-Hsiung Kuo. *Introduction to Stochastic Integration*. Universitext. New York: Springer-Verlag, 2006. ISBN: 978-0-387-28720-1.

[Man67] Benoit B. Mandelbrot. "How Long Is the Coast of Britain? Statistical Self-Similarity and Fractional Dimension". In: *Science* 156.3775 (May 1967), pp. 636–638. ISSN: 0036-8075, 1095-9203.

[Man82] Benoit B. Mandelbrot. *The Fractal Geometry of Nature*. Vol. 1. W. H. Freeman, 1982.

[MF03] Benoit B. Mandelbrot and Michael Frame. "Fractals". In: *Encyclopedia of Physical Science and Technology (Third Edition)*. Ed. by Robert A. Meyers. New York: Academic Press, Jan. 2003, pp. 185–207. ISBN: 978-0-12-227410-7.

[MPP84] Benoit B. Mandelbrot, Dann E. Passoja, and Alvin J. Paullay. "Fractal Character of Fracture Surfaces of Metals". In: *Nature* 308.5961 (Apr. 1984), pp. 721–722. ISSN: 1476-4687.

[Mun13] James Munkres. *Topology*. Pearson New International Edition. Harlow, United Kingdom: Pearson Education Canada, 2013. ISBN: 978-1-292-03678-6.

[MV68] Benoit B. Mandelbrot and John W. Van Ness. "Fractional Brownian Motions, Fractional Noises and Applications". In: *SIAM Review* 10.4 (Oct. 1968), pp. 422–437. ISSN: 0036-1445.

[Nou12] Ivan Nourdin. *Selected Aspects of Fractional Brownian Motion*. Bocconi & Springer Series. Mailand: Springer-Verlag, 2012. ISBN: 978-88-470-2822-7.

[Nua06] David Nualart. "Fractional Brownian Motion: Stochastic Calculus and Applications". In: *International Congress of Mathematicians*. Vol. 3. European Mathematical Society, 2006, pp. 1541–1562.

[Rev13] Pal Revesz. *Random Walk In Random And Non-Random Environments*. Third Edition. World Scientific, Mar. 2013. ISBN: 978-981-4447-52-2.

[SVR95] Jean Schmittbuhl, Jean-Pierre Vilotte, and Stéphane Roux. "Reliability of Self-Affine Measurements". In: *Physical Review E* 51.1 (Jan. 1995), pp. 131–147.