Un tour d'horizon des fractales

César Almecija,
Thomas Baroux,
Amine Cherif Haouat,

Lycée Privé Sainte Geneviève

23 juin 2019

Table des matières

Chapitre 1

Introduction et entrée en matière : autosimilarité et dimensions

1.1 Éviction d'une idée préconçue : les fractales sont des figures géométriques autosimilaires.

La conception commune d'une fractale est cette vision d'une forme géométrique qui est constituée de *plus petites copies d'elle-même*. C'est le cas par exemple du flocon de Koch, constitué de 4 morceaux qui sont en fait des plus petites copies de lui-même, ou encore du triangle de Sierpinski, constitué de 3 répliques de ce dernier.

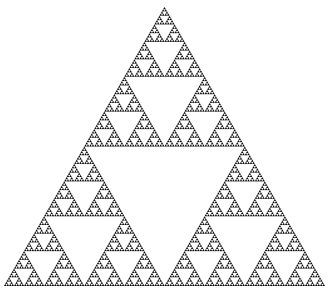


Figure 1. *Le triangle de Sierpinski, un exemple de fractale auto-similaire*

Cependant, c'est une vision assez *minimaliste*, voire assez fausse du concept de fractales tel qu'il était perçu par Mandelbrot, un des pères de cette notion. En effet, le but initial était d' "estimer la rugosité des formes géométriques". L'exemple le plus connu est donc celui du tracé des côtes du Royaume-Uni, car plus la figure est regardée de près, plus les détails apparaissent.

En clair, **une fractale n'est pas simplement une figure autosimilaire**.

1.2 Notion de dimension

Intéressons-nous maintenant la notion de **dimension**. En effet, il est assez aisé de parler de dimension lorsque nous nous référons au plan (*dimension 2*) ou à l'espace (*dimension 3*). Mathématiquement, la dimension (d'un espace vectoriel) se définit comme la taille des familles qui forment une base de cet espace (encore faut-il que cette famille soit de taille finie, sans quoi on ne peut parler de dimension).

Mais lorsque nous parlons de fractales, nous avons souvent à faire à des dimensions non-entières (environ 1.585 pour le triangle de Sierpinski, 1.262 pour le flocon de Koch, $\cdots$). Mais alors, que signifient ces **dimensions non entières** ?

C'est ici une autre approche de la notion de dimension, différente de la dimension vectorielle, qui va donc nous intéresser : elle se déduit de la mesure de Hausdorff et est nommée **dimension de Hausdorff**.

1.3 Approche pratique et notion de masse

Afin d'évaluer la *rugosité* de nos figures géométriques, plusieurs moyens ont été mis en place, afin de palier un défaut des outils conventionnels : il est par exemple impossible de parler de l'aire du triangle de Sierpinski car elle est nulle, ni de son périmètre car il est infini.

C'est ainsi qu'est introduite la notion de masse, assez minimaliste, mais qui fournit une bonne approximation de cette dernière. Son but est simple : considérons trois objets géométriques "classiques" : le cube, le carré et la ligne et dotons-les d'une "masse". En les rétrécissant d'un facteur $\frac{1}{2}$, la masse du cube est réduite d'un facteur $(\frac{1}{2})^3$ (il faut 8 cubes de cette dimension pour reformer le cube initial), celle du carré d'un facteur $(\frac{1}{2})^2$ (il faut 4 carrés de cette dimension pour reformer le carré initial) et celle de la "barre" d'un facteur $(\frac{1}{2})^1$ (il faut 2 barres de cette dimension pour reformer la barre initiale). Remarquons donc qu'à facteur de réduction $\frac{1}{2}$, la masse a un rapport de réduction de $(\frac{1}{2})^d$, où d semble représenter la *dimension* de notre objet géométrique.

En appliquant ce même principe au triangle de Sierpinski, on devient capable d'approximer sa dimension : on trouve 1.585.

Chapitre 2

La dimension fractale

L'objectif de ce chapitre est de formaliser le concept de dimension présenté ci-dessus.

2.1 Introduction à la topologie

2.1.1 Espaces topologiques, espaces métriques

Définition : On appelle **espace topologique** un couple (X, T) où X est un ensemble et T une famille de parties de X vérifiant :

i) $\varnothing \in T, X \in T$.

ii) Une intersection finie d'éléments de T appartient à T.

iii) Une réunion quelconque d'éléments de T appartient à T.

On appelle T **la topologie sur** X.

Définition : Soit X un ensemble non vide. Une **distance (métrique)** sur X est une application $(x, y) \to d(x, y)$ de $X \times X$ dans $\mathbb{R}_+$ telle que :

i) $d(x, y) = 0 \Leftrightarrow x = y$

ii) $\forall x, y \in X, d(x, y) = d(y, x)$

iii) $\forall x, y, z \in X, d(x, y) \leq d(x, z) + d(z, y)$ (inégalité triangulaire).

Définition : Un **espace métrique** est un couple (X, d), où d est une distance sur X.

Définition : Soit (X, d) un espace métrique. Pour x $\in$ X et $r > 0$, on définit :

i) la **boule ouverte de centre** x **et rayon** r :

$$B(x, r) = \{y \in X,\ d(y, x) < r\}$$

ii) la **boule fermée de centre** x **et rayon** r :

$$B(x, r) = \{y \in X,\ d(y, x) \leq r\}$$

Définition : Soit (X, d) un espace métrique. Par définition, une partie non-vide U de X est un **ouvert** si, pour tout $x \in U$, il existe un $r > 0$ tel que $B(x, r) \subset U$. Par définition, $\varnothing$ est un ouvert.

Proposition : Soit (X, d) un espace métrique. On a :

i) une **intersection finie d'ouverts** est **ouverte**.

ii) une **réunion quelconque d'ouverts** est **ouverte**.

Démonstration :

i) Soit $(U_1, ..., U_n)$ des ouverts de X. Soit $x \in \bigcap_{i=1}^{n} U_i$. Vu que tous les U_i sont ouverts, on peut fixer $r_1, ..., r_n$ tel que $\forall i \in [\![1, n]\!],\ B(x, r_i) \subset U_i$. Il suffit alors de considérer $B(x, r)$ avec $r := min(r_1, ..., r_n)$. En effet, $\forall i \in [\![1, n]\!],\ B(x, r_i) \subset B(x, r)$, donc $\forall i \in [\![1, n]\!],\ B(x, r_i) \subset U_i$, donc $B(x, r) \subset \bigcup_{i=1}^{n} U_i$.

ii) Soit $x \in \bigcup_{i \in I} U_i$. Alors $\exists i_0 \in I,\ x \in U_{i_0}$. Fixons-le. U_{i_0} étant ouvert, on peut fixer $r > 0$ tel que $B(x, r) \subset U_{i_0} \subset \cup_{i \in I}\ U_i$.

2.1.2 Compacité

Définition : Soit (X, d) un espace métrique. La **topologie métrique** de (X, d) est $T = \{U \subset X,\ U \text{ est un ouvert}\}$.

Cela nous permet de voir un espace métrique comme un cas particulier d'un espace topologique. On appelle les éléments d'une topologie T aussi les ouverts de l'espace (X, T).

Définition : Soit U une famille de parties d'un espace topologique (X, T). On dit que U est un **recouvrement ouvert de** X si $U \subset T$ (*ie* les éléments de U sont des ouverts) et $X = \bigcup_{A \in U} A$.

On dit que $V \subset U$ est un **sous-recouvrement fini** si V est fini (*ie* contient un nombre fini d'éléments) et $X = \bigcup_{A \in V} A$.

Définition : Un espace topologique X est **compact** si tout recouvrement ouvert de X admet un sous-recouvrement fini.

2.2 Dimension de Hausdorff

A partir de maintenant, on se munit d'un espace métrique (E, d) **compact pour définir la dimension de Hausdorff.**

L'idée derrière la définition de la dimension de Hausdorff est d'essayer de recouvrir notre espace métrique à l'aide de parties de E "contenues" dans des boules de diamètre au plus $r > 0$. Soit $r > 0$, que l'on conserve au long de ce paragraphe.

Le but final est d'observer le comportement de ces boules lorsqu'on fait tendre le diamètre limite vers 0.

Définition : Pour tout espace métrique X, on note :

$$diam(X) := sup\ \{\ d(a,b) \mid a,b \in X\}$$

Idée : On recouvre l'espace E au moyen d'une réunion dénombrable de parties notées $A_i \subset E$, tel que :

$$\forall i \in \mathbb{N},\ diam(A_i) \leq r$$

Puis on somme ces diamètres élevés à une puissance $s \geq 0$.

Définition : Soit $A \subset E, s \geq 0$. On définit la **mesure extérieure** par :

$$H_r^s(A) := \underset{D \in R_r(A)}{inf}\ \{\ \sum_{X \in D} (diam(X))^s\ \}$$

où $R_r(A)$ désigne l'ensemble des recouvrements dénombrables de A par des parties $X \in E$ tel que $diam(X) \leq r$.

Définition : Soit $A \subset E, s \geq 0$. On définit la **mesure de Hausdorff s-dimensionnelle** par :

$$H^s(A) := \underset{r \to 0}{lim} H_r^s(A)$$

Proposition : Soit $f : E \to E$ une similitude de rapport $r > 0$, c'est-à-dire $\forall x, y \in E,\ d(f(x), f(y)) = r \times d(x,y)$. Alors, pour tout $s > 0$, on a :

$$\forall A \subset E, H^s(f(A)) = r^s H^s(A)$$

Démonstration : Soit $A \subset E$ et $\varepsilon > 0$. On remarque que $f(R_\varepsilon(A)) = R_{r\varepsilon}(f(A))$. On a alors :

$$\begin{aligned} H_{r\varepsilon}^s(f(A)) &= \underset{D \in f(R_\varepsilon(A))}{inf} \sum_{X \in D} (diam\ X)^s \\ &= \underset{D' \in R_\varepsilon(A)}{inf} \sum_{X' \in D'} (diam\ f(X'))^s \\ &= r^s \times \underset{D' \in R_\varepsilon(A)}{inf} \sum_{X' \in D'} (diam\ X')^s \\ &= r^s \times H_\varepsilon^s(A) \end{aligned}$$

Puis lorsque $\varepsilon \to 0$, on a bien que $H^s(f(A)) = r^s H^s(A)$

Lemme : Soit $A \subset E$. S'il existe $s_0 > 0$ tel que $H^{s_0}(A) < +\infty$, alors $\forall s > s_0,\ H^s(A) = 0$.

Démonstration : Soit $\varepsilon > 0$ et $s > s_0$. On a :

$$\begin{aligned} H_\varepsilon^s(A) &\leq \underset{D \in R_\varepsilon(A)}{inf} \sum_{X \in D} \varepsilon^{s-s_0} (diam\ X)^{s_0} \\ &= \varepsilon^{s-s_0} \underset{D \in R_\varepsilon(A)}{inf} \sum_{X \in D} (diam\ X)^{s_0} \\ &= \varepsilon^{s-s_0} H_\varepsilon^{s_0}(A) \end{aligned}$$

(NB : La première inégalité se déduit du fait que la mesure de Hausdorff est une mesure métrique, aspect **non abordé** dans cette partie)

Comme $H^{s_0}_\varepsilon(A) \to H^{s_0}(A) < \infty$ quand $\varepsilon \to 0$, il vient, en faisant tendre ε vers 0 que $H^s_\varepsilon(A) = 0$, ce qui achève la démonstration.

Théorème : Soit $A \subset E$. Il existe un unique $d \in \mathbb{R}_+ \cup \{+\infty\}$ tel que :
i) $H^s(A) = +\infty$ pour tout $s < d$,
ii) $H^s(A) = 0$ pour tout $s > d$.
On appelle **dimension de Hausdorff de A** ce réel d et on le note $dim_H A$. De manière équivalente :

$$d = sup\{s \in \mathbb{R}_+ | H^s(A) = +\infty\} = inf\{s \in \mathbb{R}_+ | H^s(A) = 0\}$$

Démonstration :

1) Existence : posons $d := inf\{s \geq 0, H^s(A) < +\infty\}$.
   - Par définition, si $s < d$, $H^s(A) = +\infty$.
   - Si $s < d$, alors : $\exists s_0 \in]d, s[$, $H^{s_0}(A < +\infty)$. Puis par le lemme précédent, $H^s(A) = 0$.

2) Unicité : elle se déduit de l'unicité de l'inf (ou du sup).

Chapitre 3

Le flocon de Koch

On se propose ici de construire une fractale bien connue, le **flocon de Koch**. On réalisera une approche théorique et informatique (grâce à un programme Python).

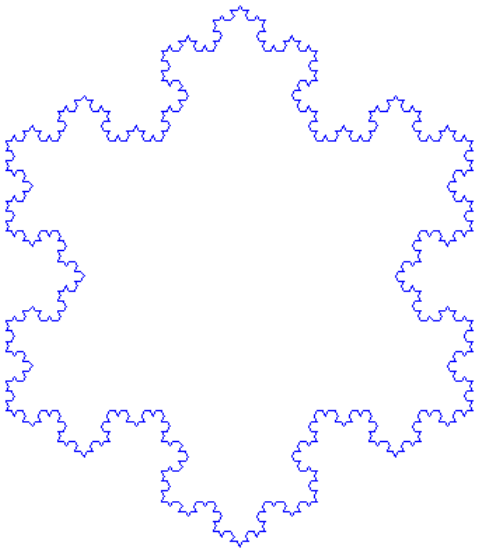


Figure 2. *Le flocon de Koch, affiché ici entièrement (dans cette partie n'est étudiée que la partie supérieure du flocon)*

L'idée est de construire une suite de flocons par récurrence, et de définir le flocon de Koch comme limite uniforme de cette suite de fonctions.

3.1 Construction des flocons par récurrence

Construisons d'abord les "points de passage" de notre flocon, qui seront donc construits par récurrence, chaque fois par rapport aux points de passage du flocon de degré de précision inférieur.

- Pour $n = 0$, il suffit juste de relier les 2 points (0,1).

- Pour $n = 1$, on ajoute 3 points : on se retrouve avec $(0, \frac{1}{3}, z, \frac{2}{3}, 1)$ où z est l'image de $\frac{2}{3}$ par la rotation d'angle $\frac{\pi}{3}$ et de centre $\frac{1}{3}$, *ie* :

$$z = \frac{1}{3} + e^{\frac{i\pi}{3}}(\frac{2}{3} - \frac{1}{3})$$

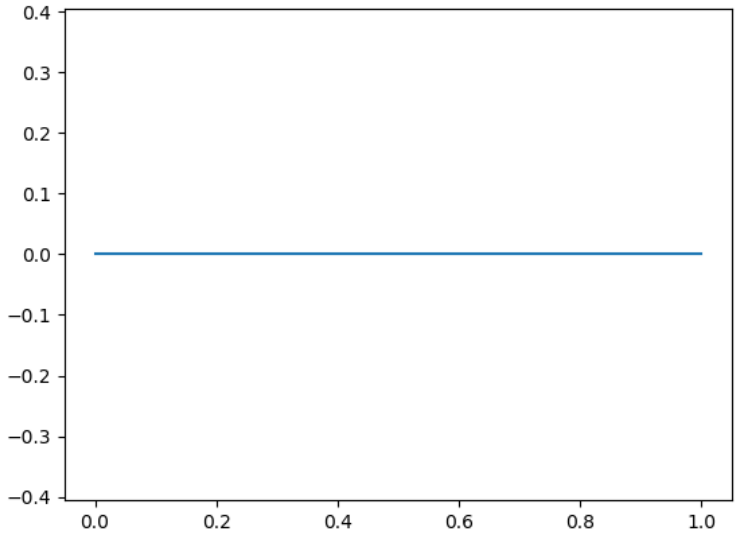


Figure 3. *Flocon initial, de degré de précision 0*

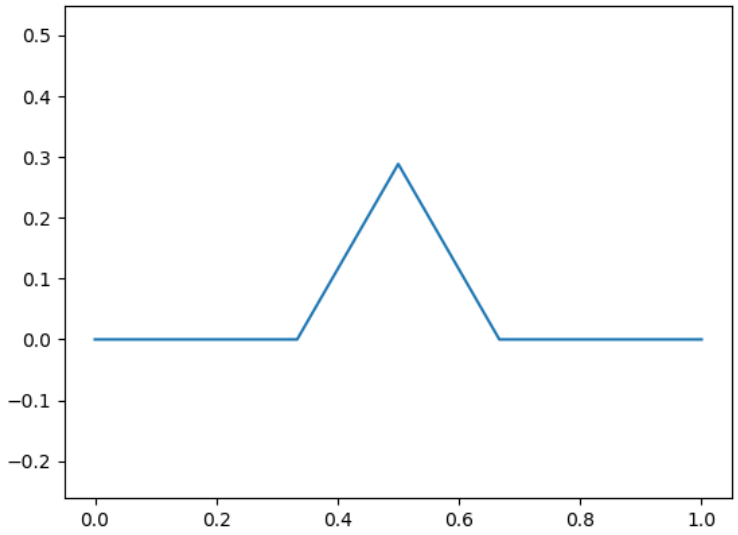


Figure 4. *Flocon de degré de précision 1*

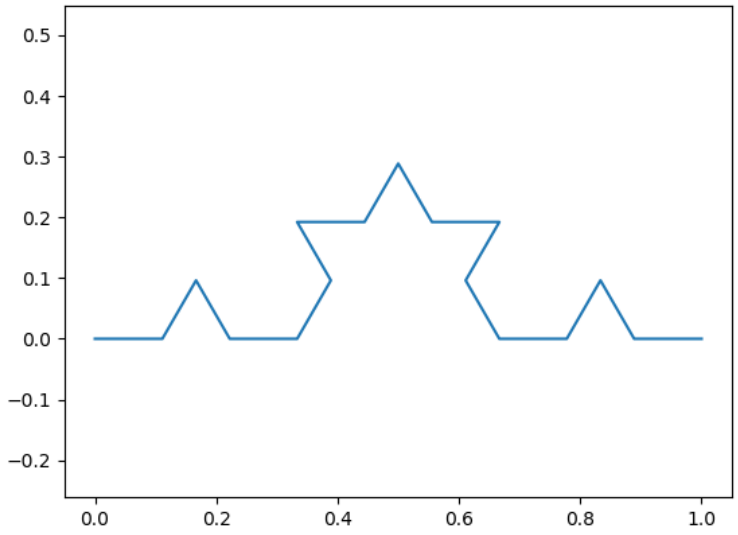


Figure 5. *Flocon de degré de précision 2*

On voit ainsi plusieurs triangles équilatéraux se former les uns sur les autres. Les programmes Python correspondants à ces images sont présent dans **l'annexe 1**.

On définit ainsi une suite de subdivisions $(\sigma_n)_{n\in\mathbb{N}}$ de la manière suivante :

$$\begin{cases} \sigma_0 = (0,1) \\ \forall n \in \mathbb{N},\ \forall i \in [\![0, |\sigma_n| - 1]\!],\ z_{n+1,4i} = z_{n,i} \\ \forall n \in \mathbb{N},\ \forall i \in [\![0, |\sigma_n| - 2]\!],\ z_{n+1,4i+1} = \frac{2}{3} z_{n,i} + \frac{1}{3} z_{n,i+1} \\ \forall n \in \mathbb{N},\ \forall i \in [\![0, |\sigma_n| - 2]\!],\ z_{n+1,4i+3} = \frac{1}{3} z_{n,i} + \frac{2}{3} z_{n,i+1} \\ \forall n \in \mathbb{N},\ \forall i \in [\![0, |\sigma_n| - 2]\!],\ z_{n+1,4i+2} = z_{n+1,4i+1} + e^{\frac{i\pi}{3}} (z_{n+1,4i+3} - z_{n+1,4i+1}) \end{cases}$$

où $|\sigma_n|$ désigne la taille de la subdivision σ_n et $z_{k,j}$ désigne le j-ème élément de la subdivision σ_k (en supposant que la numérotation des subdivisions commence à 0).

Remarque : On montre par un récurrence immédiate que

$$\forall n \in \mathbb{N},\ |\sigma_n| = 4^n + 1,$$

ce qui justifie le choix du découpage de [0,1] pour chaque f_n.

Puis on définit une suite de fonctions $(f_n)_{n\in\mathbb{N}}$ de $[0,1]$ dans $\mathbb{C}$ de la manière suivante :

$$\forall n \in \mathbb{N} \begin{cases} \forall i \in [\![0, 4^n]\!],\ f_n(\frac{i}{4^n}) = z_{n,i} \\ \\ \forall i \in [\![0, 4^n - 1]\!],\ \forall t \in]\frac{i}{4^n}, \frac{i+1}{4^n}[,\ f_n(t) = 4^n \times (f_n(\frac{i+1}{4^n}) - f_n(\frac{i}{4^n})) \times (t - \frac{i}{4^n}) \ +\ f_n(\frac{i}{4^n}) \end{cases}$$

3.2 Convergence uniforme et continuité : critère de Cauchy uniforme

Lemme : $\forall n \in \mathbb{N}^*,\ ||f_n - f_{n-1}||_\infty = \left(\frac{1}{2\sqrt{3}}\right)\left(\frac{1}{3}\right)^n$

Démonstration : soit $n \in \mathbb{N}^*$. Soit $t \in [0,1]$. Fixons $i \in [\![0, 4^{n-1} - 1]\!]$, tel que $t \in [\frac{4i}{4^n}, \frac{4i+4}{4^n}]$.

- Supposons que $t \in [\frac{4i}{4^n}, \frac{4i+1}{4^n}] \cup [\frac{4i+3}{4^n}, \frac{4i+4}{4^n}]$: on a $f_n(t) = f_{n-1}(t)$.
- Supposons que $t \in [\frac{4i+1}{4^n}, \frac{4i+3}{4^n}]$. Considérons $\mathbb{R}^2$ muni de sa structure d'espace affine (de direction lui-même) et d'origine :

$$O := (Re(f_n(z_{n,4i+1})), Im(f_n(z_{n,4i+1}))),$$

et fixons alors le repère défini par $e_1 := f_n(z_{n,4i+3}) - f_n(z_{n,4i+1})$ et e_2 tel que $z_{e_2} = e^{\frac{i\pi}{2}} z_{e_1}$. Par construction, (e_1, e_2) est une base orthogonale de $\mathbb{R}^2$, donc (O, e_1, e_2) est une repère orthogonal de $\mathbb{R}^2$ vu comme espace affine.

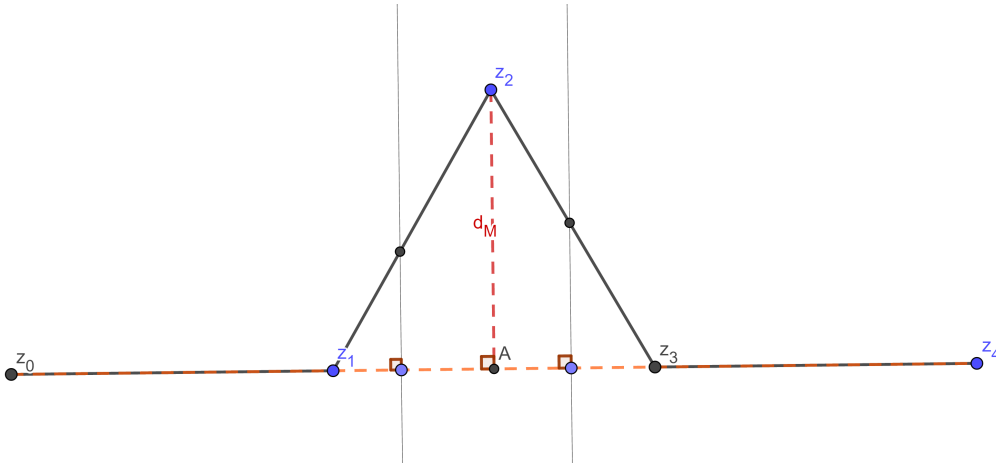


Figure 6. *Détail d'une partie du flocon*

Il vient $|f_n(t) - f_{n-1}(t)| \leq d_M := |f_n(\frac{4i+2}{4^n}) - f_{n-1}(\frac{4i+2}{4^n})|$. Or par construction, le triangle formé par les points d'affixes $f_n(z_{n,4i+1})$, $f_n(z_{n,4i+2})$ et $f_n(z_{n,4i+3})$ est équilatéral, d'où $d_M = (\frac{1}{2\sqrt{3}})|z_{n,4i+3} - z_{n,4i+1}| = (\frac{1}{2\sqrt{3}})(\frac{1}{3})^n$ (récurrence immédiate).

Définition : Par le lemme, on montre que

$$\forall n \in \mathbb{N}^*,\ \forall t \in [0,1], |f_n(t) - f_{n-1}(t)| \leq (\frac{1}{2\sqrt{3}})(\frac{1}{3})^n$$

Par le **critère de Cauchy uniforme**, on en déduit que $(f_n)_{n\in\mathbb{N}}$ converge uniformément vers une fonction $f : [0,1] \to \mathbb{C}$. C'est cette fonction que l'on nommera le **flocon de Koch**.

De plus, toujours par convergence uniforme, on retrouve que cette fonction est de plus **continue**.

3.3 Calcul pratique des dimensions

Il est clair que l'approche rigoureuse présentée ci-haut définissant la dimension de Hausdorff est difficile à mettre en place : essayer une infinité de recouvrements possibles puis trouver celui qui convient semble en première approche inaccessible.

C'est pourquoi nous allons approcher le problème de manière plus accessible. Avec Python, il nous est possible, en approximant nos recouvrements par des carrés, de quadriller le plan et de compter le nombre de carrés "touchés par la courbe". En réitérant le processus avec un quadrillage de plus en plus fin et en continuant de compter le nombre de carreaux touchés par la courbe, on peut tracer le graphe (finesse du quadrillage, nombre de carreaux touchés), dont la forme est grossièrement régie par une équation du type :

$$N = cs^d,$$

avec d la dimension recherchée. En passant au logarithme, on obtient une fonction de la forme $log(N) = log(c) + d \times log(s)$, et en évaluant le coefficient directeur, on retrouve le résultat cherché.

Le programme Python (dont le code figure en **annexe 2**) a ainsi fourni cette courbe $log(N) = f(log(s))$, où N est le nombre de carreaux touchés et s le facteur de réduction du quadrillage :

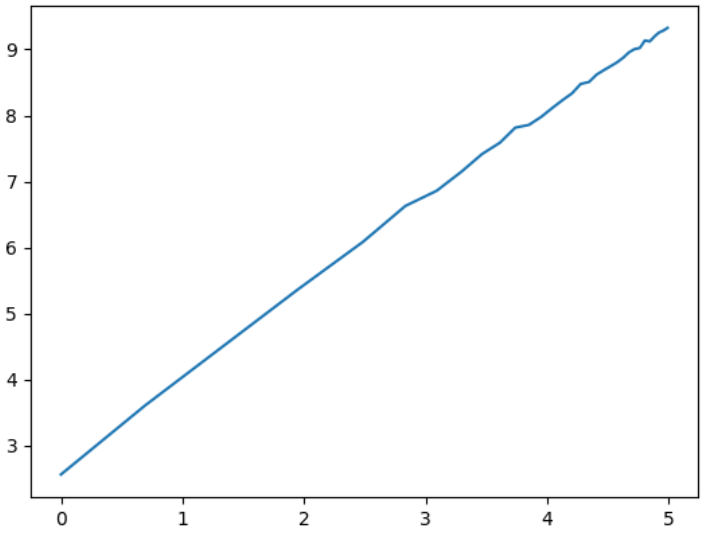


Figure 7. *La courbe obtenue par simulation informatique*

En évaluant le coefficient directeur de cette droite, on obtient bien la valeur d recherchée

Mais alors comment justifie-t-on la validité de cette méthode ?

Et bien justement, elle se base sur une manière plus théorique de calculer la dimension du flocon de Koch. Regardons-le à une certaine distance, puis appliquons-lui une homothétie de rapport 3. On retrouve exactement le même objet, composé de 4 répliques exactes du flocon avant homothétie.

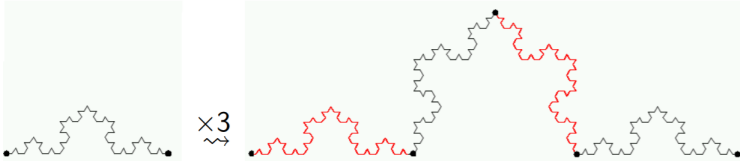


Figure 8. *Mise en évidence de l'auto-similarité du flocon de Koch*

La dimension semble alors vérifier l'équation :

$$3^d = 4$$

soit encore $d = \dfrac{ln(4)}{ln(3)} = 1,26185950714...$

Chapitre 4

L'ensemble de Mandelbrot

4.1 Propriétés géométriques

L'ensemble de Mandelbrot $\mathscr{M}$ est l'ensemble des $c \in \mathbb{C}$ tels que la suite $z_c \in \mathbb{C}^{\mathbb{N}}$ définie ainsi par récurrence soit bornée :

$$\begin{cases} z_{c_0} = 0 \\ \forall n \in \mathbb{N}, z_{c_{n+1}} = z_{c_n}^2 + c \end{cases}$$

En identifiant le plan et $\mathbb{C}$, nous pouvons représenter cet ensemble ainsi :

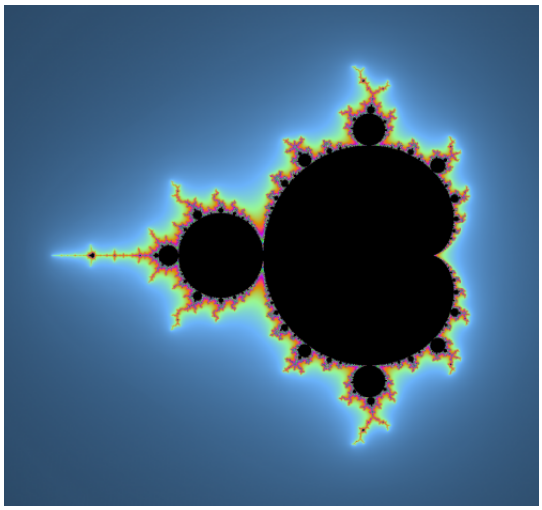


Figure 9. *L'ensemble de Mandelbrot*

Les zones noires correspondent aux points dont l'affixe appartient à l'ensemble de Mandelbrot, tandis que les zones colorées correspondent aux points dont l'affixe n'appartient pas à l'ensemble de Mandelbrot. La couleur est une convention permettant d'illustrer les différentes vitesses de divergence des suites associées à chaque point appartenant à $\mathbb{C}\backslash\mathscr{M}$. La programmation d'un algorithme permettant de représenter l'ensemble (comme celui présenté plus loin) peut sembler complexe, puisque nous n'avons aucun moyen en apparence de

prédire l'évolution de la suite des modules des termes de la suite associée à chaque complexe.

Pour faciliter ce travail, nous pouvons montrer un premier résultat, selon lequel l'existence d'un terme de module strictement supérieur à 2 dans la suite associée au nombre implique que la suite des modules tend vers $+\infty$.

Proposition : Soit $c \in \mathbb{C}$. Soit z_c la suite associée. On a

$$(\exists k \in \mathbb{N}, |z_{c_k}| > 2) \Rightarrow (|z_{c_n}| \longrightarrow +\infty)$$

Démonstration : Soit $c \in \mathbb{C}$. Cherchons les racines dans $\mathbb{R}$ de l'équation :

$$x^2 = x + |c|$$

Le discriminant vaut $\Delta = 1 + 4|c| > 0$.

Les deux solutions sont donc $\dfrac{1+\sqrt{1+4|c|}}{2}$ et $\dfrac{1-\sqrt{1+4|c|}}{2}$
Notons α la plus grande des deux. On a $\alpha \geq 1$. Posons à présent la suite (β_n) définie par :

$$\forall n \in \mathbb{N}, \beta_n := |z_{c_n}| - \alpha$$

Puis, soit $n \in \mathbb{N}$, on a :

$$\begin{aligned} \alpha + \beta_{n+1} &= \left|z_{c_{n+1}}\right| \\ &= \left|z_{c_n}^2 + c\right| \\ &= \left|z_{c_n}^2 - (-c)\right| \\ &\geq |z_{c_n}|^2 - |c| \end{aligned}$$

On a donc :

$$\begin{aligned} \beta_{n+1} &\geq (\alpha + \beta_n)^2 - \alpha^2 \\ &\geq 2\alpha\beta_n \end{aligned}$$

Ainsi, il suffit qu'il existe k $\in \mathbb{N}$ tel que $\beta_k > 0$ (*ie* qu'il existe k $\in \mathbb{N}$ tel que $|z_{c_k}| > \alpha$) pour avoir $\beta_n \longrightarrow +\infty$ (car on a $2\alpha > 0$), et donc $|z_{c_n}| \longrightarrow +\infty$. Or, dans le cas où $|c| > 2$ on a :

$$\begin{aligned} &4|c|^2 - 4|c| > 4|c| \text{ (car } 4|c| > 0), \\ \text{donc } &(2|c| - 1)^2 > 1 + 4|c|, \\ \text{donc } &2|c| > 1 + \sqrt{1+4|c|} \text{ (stricte croissance de la racine)}, \\ \text{donc } &|c| > \alpha. \end{aligned}$$

Or $|c| = |z_{c_1}|$, donc quand $|c| > 2$, il suffit de poser $k = 1$.

Dans le cas où $|c| \leq 2$, on a désormais :

$$\begin{aligned} |c| &\leq 2, \\ \text{donc } 1 + 4\,|c| &\leq 9, \\ \text{donc } \sqrt{1 + 4\,|c|} &\leq 3, \\ \text{donc } \alpha &\leq 2. \end{aligned}$$

Donc s'il existe k tel que $|z_{c_k}| > 2$, alors nécessairement $|z_{c_k}| > \alpha$, ce qui achève la démonstration.

Nous pouvons à présent énoncer quelques résultats géométriques sur $\mathscr{M}$.

Corollaire : $\mathscr{M}$ est inclus dans le disque de rayon 2.

Démonstration : Nous avons montré que s'il existe $k \in \mathbb{N}$ tel que $|z_{c_k}| > 2$, alors on a que $|z_{c_n}| \to +\infty$.

Notons D le disque de rayon 2. On a :

$$\forall c \in \mathbb{C} \backslash D, |c| > 2 \text{ (par définition de } D).$$

On a donc $\forall c \in \mathbb{C} \backslash D, |z_{c_1}| > 2$ donc $c \notin \mathscr{M}$.

Théorème [symétrie axiale] : $\mathscr{M}$ est **symétrique** par rapport à l'axe des abscisses. En clair, nous avons que : $\forall c \in \mathbb{C}, c \in \mathscr{M} \Leftrightarrow \overline{c} \in \mathscr{M}$.

Démonstration : Effectuons la preuve par récurrence. Soit $c \in \mathbb{C}$, montrons que $\forall n \in \mathbb{N}, z_{\overline{c}_n} = \overline{z_{c_n}}$, ce qui suffira à conclure par invariance du module par conjugaison. Posons $\forall n \in \mathbb{N}, H_n :=$ "$\forall c \in \mathbb{C}, z_{\overline{c}_n} = \overline{z_{c_n}}$".

Initialisation : Soit $c \in \mathbb{C}$, on a $z_{\overline{c}_0} = 0 = \overline{z_{c_0}}$

Hérédité : Soit $n \in \mathbb{N}$ tel que H_n vraie. Montrons que H_{n+1} est vraie.

$$\begin{aligned} z_{\overline{c}_{n+1}} &= z_{\overline{c}_n}^2 + \overline{c} \\ &= \overline{z_{c_n}}^2 + \overline{c} \text{ (par hypothèse de récurrence)} \\ &= \overline{z_{c_n}^2 + c} \text{ (car } z \mapsto \overline{z} \text{ est un morphisme d'anneaux)} \\ &= \overline{z_{c_{n+1}}} \end{aligned}$$

Ce qui achève la démonstration.

Par ailleurs, s'il est difficile de prévoir l'appartenance à $\mathscr{M}$ ou non d'un complexe de D en général, il est revanche plus facile de la déterminer dans le cas où le nombre est réel.

Théorème [Intersection avec $\mathbb{R}$] : On a $\mathscr{M} \cap \mathbb{R} = [-2, \frac{1}{4}]$.

Démonstration : 1) Montrons dans un premier temps par récurrence que $[-\frac{1}{4}, \frac{1}{4}] \subset \mathscr{M} \cap \mathbb{R}$.

Posons $\forall n \in \mathbb{N}, H_n =$ "$\forall c \in [-\frac{1}{4}, \frac{1}{4}],\ |z_{c_n}| \leq \frac{1}{2}$".

Initialisation : Soit $c \in [-\frac{1}{4}, \frac{1}{4}]$. On a $|z_{c_0}| = |0| = 0 \leq \frac{1}{2}$

Hérédité : Soit $n \in \mathbb{N}$ tel que H_n vraie. Montrons que H_{n+1} est vraie. Soit $c \in [-\frac{1}{4}, \frac{1}{4}]$:

$$\begin{aligned} \left|z_{c_{n+1}}\right| &= \left|z_{c_n}^2 + c\right| \\ &\leq \left|z_{c_n}^2\right| + |c| \text{ (par l'inégalité triangulaire)} \\ &= |z_{c_n}|^2 + |c| \\ &\leq \frac{1}{2}^2 + \frac{1}{4} \text{ (par hypothèse de récurrence)} \\ &= \frac{1}{2} \end{aligned}$$

Ce qui achève la récurrence.

2) Montrons à présent le cas où $-2 \leq c < -\frac{1}{4}$. Procédons là encore par récurrence.

Posons $\forall n \in \mathbb{N}, H_n :=$ "$\forall c \in [-2, -\frac{1}{4}[,\ |z_{c_n}| \leq |c|$".

Initialisation : Soit $c \in [-2, -\frac{1}{4}[$. On a bien $0 \leq |c|$.

Hérédité : Soit $n \in \mathbb{N}$ tel que H_n vraie. Montrons que H_{n+1} est vraie. Soit $c \in [-2, -\frac{1}{4}[$.

On a :

$$\begin{aligned} \left|z_{c_{n+1}}\right| &= \left|z_{c_n}^2 + c\right| \\ &\leq |z_{c_n}|^2 + |c| \\ &\leq |c|^2 + |c| \text{ (par hypothèse de récurrence)} \\ &= |c|\,|1 + c| \\ &\leq |c| \end{aligned}$$

Ce qui achève la récurrence.

3) Reste enfin le cas où $c > \frac{1}{4}$.

$$\begin{aligned} \text{Soit } f : \mathbb{R} &\to \mathbb{R} \\ x &\mapsto x^2 - x \end{aligned}$$

On a $\forall x \in \mathbb{R}, f'(x) = 2x - 1$, donc f admet un minimum en $\frac{1}{2}$. Puisque le coefficient dominant vaut 1, on a que l'extremum est un minimum. Enfin, $f(\frac{1}{2}) = -\frac{1}{4}$.

Soit $(n, c) \in \mathbb{N} \times \mathbb{R}$:

$$\begin{aligned} z_{c_{n+1}} - z_{c_n} &= z_{c_n}^2 + c - z_{c_n} \\ &\geq c - \frac{1}{4} (\text{car } \forall p \in \mathbb{N}, z_{c_p} \in \mathbb{R}) \end{aligned}$$

Ainsi, si $c > \frac{1}{4}$, on a que $\forall n \in \mathbb{N}, z_{c_n} \geq n(c - \frac{1}{4})$. D'après le théorème de minoration, on aurait alors $|z_{c_n}| \to +\infty$

Par conséquent, on a bien :

$$\mathscr{M} \cap \mathbb{R} = [-2, \frac{1}{4}]$$

4.2 Propriétés topologiques

4.2.1 Définitions

Fonction analytique : On dira d'une fonction qu'elle est **analytique** lorsque lorsqu'elle est développable en série entière au voisinage de chacun des points de son ensemble de définition D, c'est-à-dire que :

$$\forall x_0 \in D,\ \exists U \in V_{x_0},\ \exists \left(a_n\right)_{n\in\mathbb{N}} \in \mathbb{C}^{\mathbb{N}}, \forall t \in U, f(t) = \sum_{n=0}^{+\infty} a_n \left(t - x_0\right)^n$$

Théorème [Combinaison linéaires de fonctions analytiques] : Toute combinaison linéaire de fonctions analytiques est analytique.

Démonstration : Soit D l'ensemble de définition commun à f et g analytiques. Soit $x_0 \in D$ et U un voisinage de x_0. Fixons $\left(\left(a_n\right)_{n\in\mathbb{N}}, \left(b_n\right)_{n\in\mathbb{N}}\right) \in \left(\mathbb{C}^{\mathbb{N}}\right)^2$ telles que :

$$\forall x \in U, f(x) = \sum_{n=0}^{+\infty} a_n \left(x - x_0\right)^n \text{ et } g(x) = \sum_{n=0}^{+\infty} b_n \left(x - x_0\right)^n$$

Soit $(\lambda, \mu) \in \mathbb{C}^2$,
Posons $\forall n \in \mathbb{N},\ c_n := \lambda a_n + \mu b_n$.
Il vient que

$$\forall x \in U,\ \left(\lambda f + \mu g\right)(x) = \sum_{n=0}^{+\infty} c_n \left(x - x_0\right)^n$$

(qui converge par combinaison linéaire sur des séries).

Théorème [de Borel-Lebesgue] : Tout ensemble borné et fermé est compact.

*Démonstration :*Soit $(a,b) \in \mathbb{R}^2$ tels que $a < b$. Montrons que $[a,b]$ est compact.

Soit $(U_i)_{i\in I}$ un recouvrement ouvert de $[a,b]$.

Posons $E := \Big\{ x \in [a,b] \mid \exists J \subset I \text{ fini }, (U_j)_{j\in J} \text{ soit un recouvrement fini de } [a,x] \Big\}$.

Montrons que E est non-vide ; il suffit de montrer que $a \in E$. $(U_i)_{i\in I}$ est un recouvrement de $[a,b]$. Fixons $i \in I$ tel que $a \in U_i$. Il vient que $(U_k)_{k\in\{i\}}$ est un recouvrement de $[a,a]$ et que $\{i\}$ est fini (de cardinal 1). Donc $a \in E$, donc E est non vide.

De plus, $\forall x \in E,\ x \leq b$, donc E est majoré. Fixons donc $M = \sup(E)$.

Fixons $k \in I$ tel que $M \in U_k$ et $\varepsilon > 0$ tel que :

$$P := [M-\varepsilon, M+\varepsilon] \cap [a,b] \subset U_k.$$

Fixons c,d tels que $P = [c,d]$. M étant la borne supérieure de E, on peut fixer $y \in P \cap E$. En rajoutant U_k au recouvrement fini de $[a,y]$ (licite car $y \in E$), on obtient un recouvrement de $[a,d]$.

Or, puisque $d = \max(P \cap E)$, on a $d = \min(M+\varepsilon, b)$. Puisque $M \geq d$, nécessairement, on a $d = b$. On a donc un recouvrement fini de $[a,b]$.

Définition [Sphère de Riemann] : On nomme **sphère de Riemann** la sphère correspondant à une projection de $\mathbb{C} \cup \{\infty\}$, que l'on notera $\bar{\mathbb{C}}$. On la représente et on la construit ainsi :

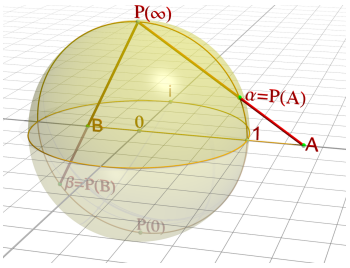


Figure 10. *Représentation de la sphère de Riemann*

4.2.2 La connexité de l'ensemble de Mandelbrot

Lemme [de Böttcher-Fatou] : Soit $k \geq 2$. Soit $(a_n)_{n\in\mathbb{N}} \in \mathbb{C}^{\mathbb{N}}$. Supposons que $z \mapsto z^k + a_{k+1}z^{k+1} + \cdots$ est analytique au voisinage de 0. Posons :

$$\forall n \in \mathbb{N},\ \phi_n : z \mapsto (f^n(z))^{\frac{1}{k^n}} = z + a_1 z^2 + \cdots$$

Alors dans un voisinage U de 0, la fonction $\phi : U \to B_r(0)$ définie comme étant égale à $\lim_{n\to\infty} \phi_n$ vérifie :

i) $\phi \circ f \circ \phi^{-1} = .^k$

ii) $\phi(0) = 0$ et $\phi'(0) = 1$

Démonstration : Montrons que la suite $(\phi_n)_{n\in\mathbb{N}}$ converge uniformément. Posons :

$$h := z \mapsto \log\left(\frac{f(z)^{\frac{1}{k}}}{z}\right)$$

On a $f(z)^{\frac{1}{k}} \sim z$ au voisinage de 0 donc $\dfrac{f(z)^{\frac{1}{k}}}{z} \sim 1$ donc h est bien définie. Elle est analytique par composition et quotient par l'identité.

Fixons donc $C \in \mathbb{R}_+^*$ tel que $\forall z \in U,\ |h(z)| \leq C|z|$. Quitte à diminuer U, on a $f(U) \subset U$ et $|f(z)| \leq |z|$ car $k \geq 2$ donc $|f(z)| = o(|z|)$ au voisinage de 0.

Ecrivons à présent ϕ sous la forme du produit infini suivant, en posant :

$$\forall z \in U,\ \phi(z) = z \times \frac{\phi_1(z)}{z} \times \frac{\phi_2(z)}{\phi_1(z)} \times \frac{\phi_3(z)}{\phi_2(z)} \cdots$$

.

Le produit converge car $\forall z \in U,\ \sum_{n=0}^{\infty} \log \dfrac{\phi_{n+1}(z)}{\phi_n(z)}$ converge uniformément.

Soit $z \in U$. On a :

$$\begin{aligned}
\left|\log \frac{\phi_{n+1}(z)}{\phi_n(z)}\right| &= \left|\log\left[\frac{(f\circ f^n(z))^{\frac{1}{k}}}{f^n(z)}\right]^{\frac{1}{k^n}}\right| \\
&= \frac{1}{k^n}.|h(f^n(z))| \\
&\leq \frac{1}{k^n} C.|f^n(z)| \\
&\leq \frac{C.|z|}{k^n}
\end{aligned}$$

Or, $\sum_{n=0}^{\infty} \dfrac{1}{k^n}$ converge car $k \geq 2 > 1$. Donc ϕ est bien définie. Il vient ensuite que $\phi(0) = 0$ et $\phi'(0) = 1$. On constate de plus que $\forall z \in U,\ \phi(f(z)) = \phi(z)^k$, ce qui achève la démonstration.

Définition [Ensemble de Fatou et de Julia] L'**ensemble de Fatou** associé à un complexe $c \in \mathbb{C}$ est l'ensemble des $z \in \mathbb{C}$ tel qu'il existe un voisinage U de z tel que la suite définie par :

$$\begin{cases} z_{c_0} = z \\ \forall n \in \mathbb{N}, z_{c_{n+1}} = z_{c_n}^2 + c \end{cases}$$

ait même comportement pour tout élément de U. Il s'agit donc d'un ouvert par définition. Son complémentaire est appelé **ensemble de Julia**, qui est donc fermé par définition.

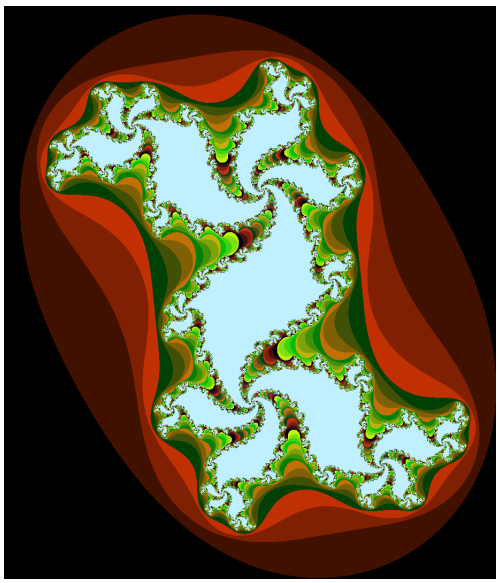


Figure 11. *Ensemble de Julia associé à* $c = 0.3 + 0.5i$

Remarque : L'ensemble de Julia correspond en réalité aux valeurs de convergence de la suite. Il est chaotique, à l'image de $\mathscr{M}$, ce qui explique son caractère fermé, et donc non stable par faibles variations. De plus, on peut démontrer que J_c est connexe si et seulement si $c \in \mathscr{M}$.

Cela permet de définir alternativement l'ensemble de Mandelbrot comme l'ensemble des $c \in \mathbb{C}$ tels que J_c est connexe.

Proposition : L'ensemble de Julia est compact.

Démonstration : Pour montrer que J_c est compact, montrons qu'il est borné et fermé, ce qui suffira à conclure.

- Il est fermé par définition, comme complémentaire de l'ensemble F_c, ouvert par définition.
- D'après le lemme précédent, pour z tel que $|z|$ soit suffisamment grande, on a que la suite tend vers $+\infty$. L'ensemble de Julia est donc borné.

Il est donc compact par le théorème de Borel-Lebesgue.

Définition [la fonction Φ] : On admettra l'existence d'une fonction, dite de *Green* et notée G, telle qu'elle permette de définir une fonction :

$$\Phi \ : \ c \ \mapsto \ G_c(c)$$

Φ est analytique sur $\bar{\mathbb{C}}\backslash M$, et on a $\Phi(z) = \lim_{n\to\infty} [f_c^n(z)]^{\frac{1}{2^n}}$. Ses propriétés permettront de conclure quand à la connexité de $\mathscr{M}$.

Toutefois, la démonstration complète de la connexité de $\mathscr{M}$ n'est pas accessible à notre niveau. En effet, elle fait appel aux notions de familles normales de fonctions, au principe de l'argument ou encore au théorème de Goursat.

4.2.3 Simple connexité de la sphère

Nous allons ici montrer que la sphère de Riemann est **simplement connexe dans l'espace**. Cela signifie que pour tout lacet de la sphère, nous pouvons fixer une fonction continue qui amène ce lacet sur un point. On se donne donc une sphère $\mathscr{S}$ de centre O de coordonnées $(0,0,0)$. Son rayon sera noté R, et on prendra $R = 1$.

Pour cela, nous allons définir un nouveau système de coordonnées sur la sphère. On considère le repère direct (O,x,y,z).

Soit $P \in \mathscr{S}$ un point de la sphère en question. Notons (x_P, y_P, z_P) ses coordonnées. Nous allons donc définir un angle $\phi \in [0,\pi]$ permettant en quelque sorte de "remplacer" z_P dans la définition de P en fonction de ses coordonnées, c'est à dire que nous allons poser une bijection entre $[-1,1]$ et $[0,\pi]$. Définissons donc :

$$\begin{aligned} \phi \ : \ [-1,1] &\to \ [0,\pi] \\ z \ &\mapsto \ \arccos(z) \end{aligned}$$

On a que ϕ est bijective comme bijection réciproque de $\cos_{[0,\pi]}^{[-1,1]}$.

Ainsi, en posant $\forall P \in \mathscr{S}, \ \phi(z_p) := \phi_P$, on peut caractériser P par les coordonnées (x,y,ϕ_P).

Cas où le lacet $\mathscr{L}$ ne recouvre pas l'ensemble de la sphère

Le lacet ne recouvrant pas l'ensemble de la sphère, nous pouvons donc par hypothèse fixer un point $\tilde{A} \in \mathscr{S}$ tel que $\tilde{A}$ n'appartienne pas au lacet. Nous allons donc fixer A le point antipodal à $\tilde{A}$, *ie* le point de coordonnées $(-x_{\tilde{A}}, -y_{\tilde{A}}, \pi - \phi_{\tilde{A}})$. Nous allons donc faire converger le lacet vers ce point A. Sans perte de généralité, supposons que ce point ait pour coordonnées $(0,0,0)$ (dans le nouveau système de coordonnées, cela correspond au plus haut point de la sphère).

Soit $P \in \mathscr{L}$. Notons d_P la demi-droite d'origine O passant par le projeté orthogonal de P sur le plan engendré par $(\overrightarrow{x}, \overrightarrow{y})$.

A présent, notons h_{P_x} la fonction qui à tout t dans $[0,1]$ associe le produit scalaire de $\overrightarrow{x}$ et de l'unique point d'intersection de d_P et du cercle dans le plan engendré par $(\overrightarrow{x}, \overrightarrow{y})$ de centre O et de rayon $\sin(\phi_P(1-t))$. Posons symétriquement h_{P_y}, et définissons la fonction f_P qui fait converger P :

$$\begin{aligned} f_P \ : \ [0,1] &\to \ \mathscr{S} \\ t &\mapsto \left(h_{P_x}(t), h_{P_y}(t), \phi_P(1-t)\right) \end{aligned}$$

Enfin, posons f, qui fait converger le lacet :

$$\begin{aligned} f \ : \ [0,1] \times \mathscr{L} &\to \mathscr{S} \\ (t, P) \ &\mapsto f_P(t) \end{aligned}$$

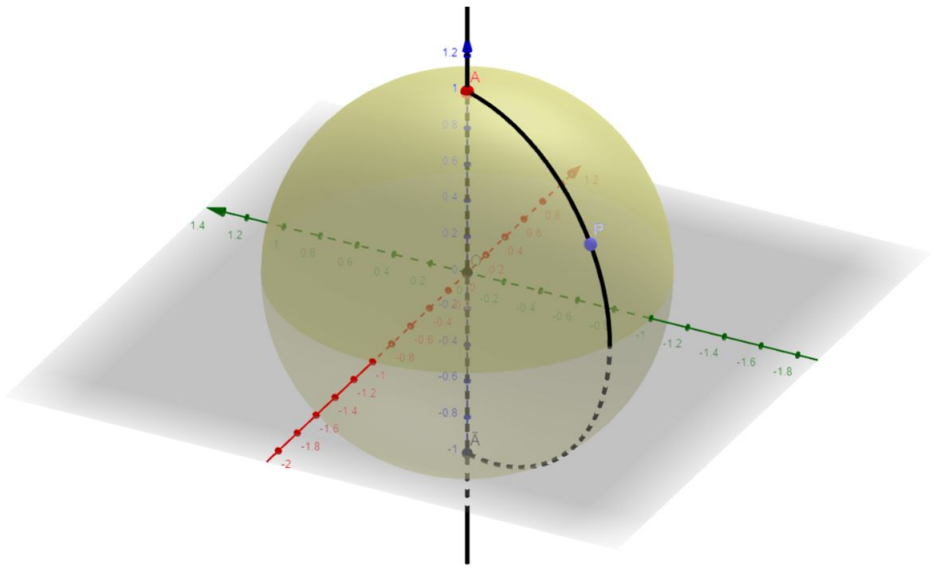


Figure 12. *Représentation de la convergence d'un point P (en bleu) vers le point A*

Explication physique : En clair, il s'agit de prendre un point du lacet n'appartenant pas à la sphère. Pour le besoin de l'explication physique, nous le placeront au "pôle Nord" de la sphère, et nous ferons converger le lacet vers le "pôle Sud" ; il est bon de noter que la méthode de convergence présentée ci-dessus faisait exactement le contraire, *ie* elle plaçait ce point au "pôle Sud" et faisait converger le lacet vers le "pôle Nord".

Nous allons ensuite lester chaque point du lacet et le lier par une barre rigide au centre de la sphère. Ensuite, nous lâchons les points, et ceux-ci vont chuter par un axe correspondant à leur méridien, pour finir tous au "pôle Sud" de la

sphère.

Une autre différence avec la "fonction de chute" f définie précédemment est que, dans le cas de f, les points atteignent tous le "pôle Sud" en même temps, ce qui ne serait pas le cas des masses ponctuelles chutant le long de la sphère (les plus proches du "pôle Sud" l'atteignent en premier).

De plus, cela explique également pourquoi nous avons besoin, dans ce cas particulier, d'un point n'appartenant au lacet en haut de la sphère (ou pour notre fonction, d'un point tel que $\phi_P = \pi$). Dans le cas de notre lacet lesté, on a que le point au sommet est dans une position d'équilibre, au même titre que le point au "pôle Sud". La différence est que l'équilibre est instable, c'est-à-dire qu'un léger décalage, même infinitésimal, entraînerait la chute du point vers le "pôle Sud". Ainsi, tous les points au voisinage épointé du sommet chuteraient, mais pas celui au sommet. Ainsi, le lacet ne rejoindrait jamais entièrement le "pôle Sud", à cause du "pôle Nord".

Afin de le faire rejoindre le "pôle Sud", il faudrait décider arbitrairement d'une direction dans laquelle le faire partir, c'est à dire choisir un méridien à suivre parmi l'infinité de méridiens auxquels le point appartient. Ainsi, en choisissant un méridien, on occasionnerait une brisure spontanée de symétrie, entraînant par la même occasion une rupture du lacet, puisque le point partirait dans une direction alors qu'un point "adjacent" partirait dans la direction opposée.

Cas où le lacet $\mathscr{L}$ recouvre l'ensemble de la sphère

L'existence de ce type de courbe mettant $[0,1]$ *en surjection avec une surface est détaillée dans le chapitre suivant.*

Notons $\mathscr{L} : [0,1] \to \mathscr{S}$ la fonction construisant le lacet. Le lacet est une fonction continue de $[0,1]$ dans $\mathbb{R}$. $[0,1]$ étant un segment, on peut appliquer le théorème de Heine. On obtient que $\mathscr{L}$ est uniformément continue. Ainsi, on a que :

$$\forall \varepsilon > 0,\ \exists \delta > 0,\ \forall (x,y) \in [0,1]^2,\ |x-y| < \delta \Rightarrow ||\mathscr{L}(x) - \mathscr{L}(y)|| < \varepsilon$$

où $||.||$ est la norme canonique de $\mathbb{R}^3$.

Fixons donc $\delta > 0$ associé à $\varepsilon = 1$. Quitte à diminuer δ, on peut ainsi partitionner $[0,1]$ en $n := \dfrac{1}{\delta}$ intervalles, chacun de largeur δ.

Ces intervalles ne peuvent recouvrir l'ensemble de la sphère pris séparément. En effet, le théorème de Heine nous assure que deux points pris dans l'image directe d'un intervalle de largeur δ ne pourront pas être éloignés l'un de l'autre d'une distance supérieure à 1. Or, 1 étant le rayon de la sphère, deux points antipodaux sont distants de $2 > 1$. Donc il ne pourra y avoir de points antipodaux dans l'image directe d'un intervalle de longueur δ. Ainsi, l'image de chaque intervalle ne remplira pas toute la sphère.

On peut ensuite projeter pour tout $i \in [\![0, n-1]\!]$, $\mathscr{L}_{[i\delta,(i+1)\delta]}$ sur l'arc de cercle reliant $\mathscr{L}_{i\delta}$ à $\mathscr{L}_{(i+1)\delta}$ en utilisant la même méthode que dans 1, puisque l'on peut utiliser un point n'appartenant pas à cette portion de lacet.

On obtient ainsi une union finie de n arcs de cercle, tous d'intérieur vide. D'après le théorème de Baire, leur union est également d'intérieur vide. A fortiori, elle ne recouvre pas toute la sphère, on peut donc lui appliquer le paragraphe précédent. En effet, il ne s'agit plus forcément d'un lacet, mais la méthode précédente n'utilise pas le fait que $\mathscr{L}(0) = \mathscr{L}(1)$.

Par conséquent, la sphère est bien **simplement connexe**.

4.3 Affichage de l'ensemble de Mandelbrot par un programme Python

Le programme Python expliqué et utilisé dans cette section pour afficher l'ensemble de Mandelbrot est présenté en **annexe 3**.

4.3.1 Explication du code et exemples

Dans le code pour afficher l'ensemble de Mandelbrot, il y a de nombreux arguments. Les quatre premiers ne sont là que pour choisir la zone que l'on souhaite afficher.

En effet, l'algorithme est assez long à faire tourner, puisqu'il y un total $N \times c$ suites à étudier, soit approximativement N^2 suites, pour une fenêtre de forme presque carrée. Par conséquent, si nous ne sommes intéressés que par une certaine portion de l'ensemble, il est important de le signaler à l'ordinateur.

Par ailleurs, s'il est possible d'affirmer avec certitude qu'une suite est non bornée lorsque son module dépasse 2 à un certain rang, il est en revanche impossible de d'affirmer qu'une suite converge en étudiant empiriquement un grand nombre de termes (à l'exception des termes réels). Pour cette raison, le paramètre T du code permet de définir le nombre de termes de la suite que nous allons calculer au maximum, pour étudier leur module.

Pour une valeur trop faible, certains points apparaîtront comme appartenant à l'ensemble, alors que le module de leur suite allait dépasser 2 pour un nombre d'itérations supérieur à T. En revanche, rentrer une valeur trop importante de T ralentira significativement les calculs, car elle augmentera le nombre de calculs de les suites liées à chaque point du plan $\mathscr{P}$. Nous allons donc montrer le résultat obtenu avec différentes valeurs de T, avec les autres paramètres constants. On prendra $N = 1000$.

- Pour $T = 5$, on obtient :

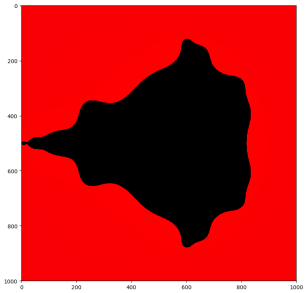


Figure 13. *Ensemble de Mandelbrot pour* $T = 5$

On obtient ainsi une image ne ressemblant que peu à l'ensemble. On note cependant que la forme générale est présente, avec quelques formes particulières.

- Pour $T = 7$, on obtient :

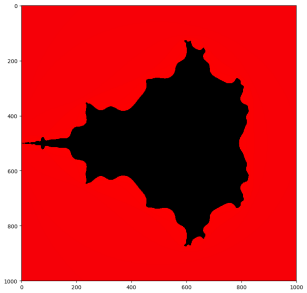
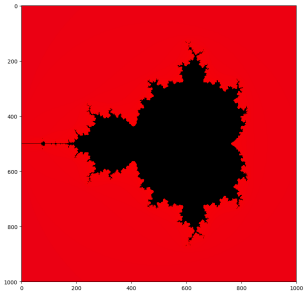
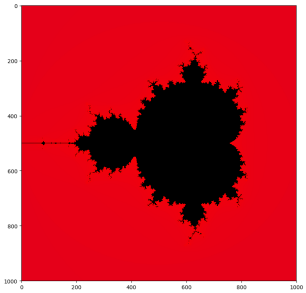


Figure 14. *Ensemble de Mandelbrot pour* $T = 7$

Si la forme est encore grossière, on commence à distinguer certaines branches. Le très faible temps de calcul (car peu d'itérations) ne peut cependant justifier une aussi faible définition.

- Pour $T = 15$ et $T = 20$, on obtient :





Figures 15 et 16. *Ensemble de Mandelbrot pour* $T = 15$ *et* $T = 20$
La majeure partie des embranchements est visible, mais ils sont encore assez grossiers. Le temps de calcul est à présent de quelques secondes.

- Pour $T = 30$, on obtient :

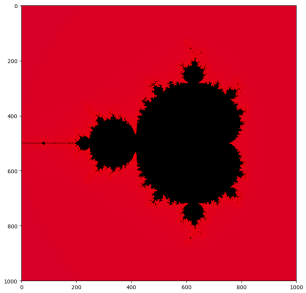


Figure 17. *Ensemble de Mandelbrot pour* $T = 30$
En se rapprochant d'un creux, les nombreux bourgeons sont visibles, et le creux est assez marqué :

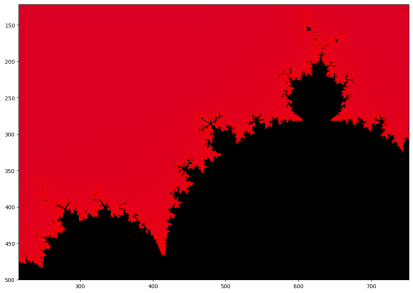


Figure 18. *Détail de l'ensemble de Mandelbrot pour* $T = 30$

- Pour $T = 50$, on obtient :

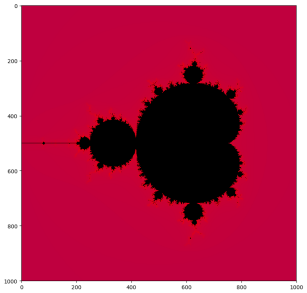


Figure 19. *Ensemble de Mandelbrot pour* $T = 50$

Et en se rapprochant :

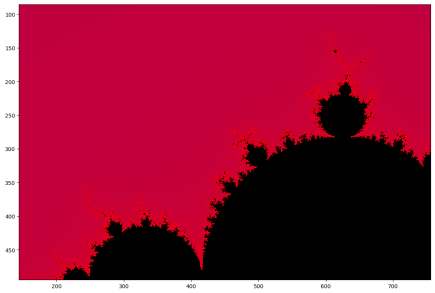
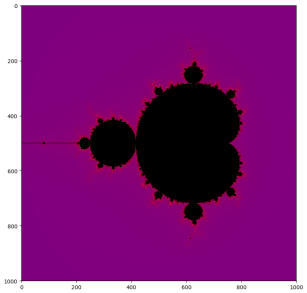
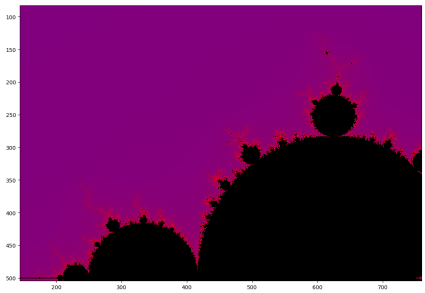


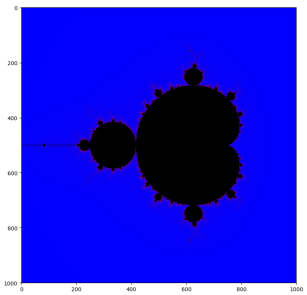
Figure 20. *Détail de l'ensemble de Mandelbrot pour* $T = 50$

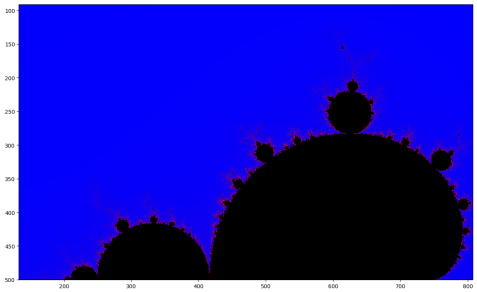
Les bourgeons sont maintenant nets. A présent, il faudrait accroître la valeur de N pour obtenir une précision supérieure. Augmenter la valeur de T n'a plus tellement de sens à présent d'un point de vue de précision, et cela rallonge inutilement les calculs.

- En effet, voici les résultats pour $T = 100$ (vue générale et agrandie) et $T = 200$ (idem) :









Figures 21 à 24. *Ensemble de Mandelbrot pour* $T = 100$ *(vue générale puis détail) puis pour* $T = 200$ *(vue générale puis détail)*

4.3.2 De la démultiplication de l'ensemble

Un argument n'a pas encore été exploité dans l'algorithme, à savoir e. Voilà différentes images obtenues pour différentes valeurs de e.

- Pour $e = 2$, obtient l'ensemble classique :

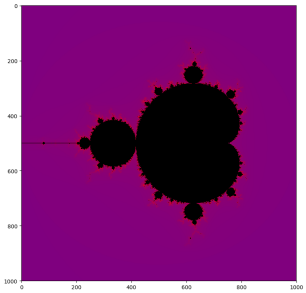


Figure 25. *Ensemble de Mandelbrot pour* $e = 2$

En revanche, pour des valeurs de e strictement supérieures à 2, nous obtenons une sorte de démultiplication de l'ensemble :

- Pour $e = 3$:

Figure 26. *Ensemble de Mandelbrot pour* $e = 3$

- Pour $e = 4$:

Figure 27. *Ensemble de Mandelbrot pour* $e = 4$

- Pour $e = 5$:

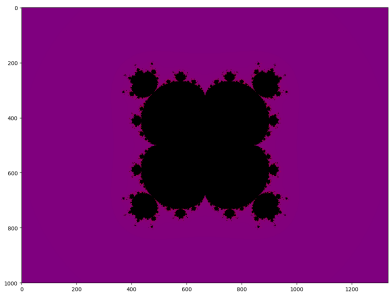


Figure 28. *Ensemble de Mandelbrot pour* $e = 5$

- Pour $e = 10$:

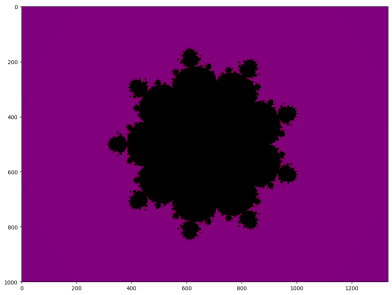


Figure 29. *Ensemble de Mandelbrot pour* $e = 10$

- Pour $e = 100$:

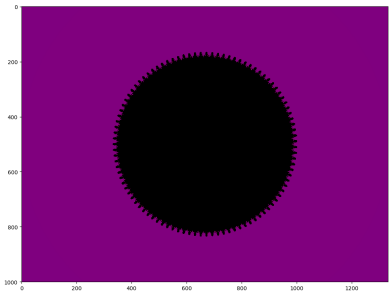


Figure 30. *Ensemble de Mandelbrot pour* $e = 100$

Bien qu'il soit déformé à chaque fois, il semblerait que nous obtenions $(e-1)$ fois l'ensemble, pour $e \geq 2$. Pour le résultat qui suit, e sera noté t pour éviter le conflit de notation avec le nombre d'Euler.

Théorème [Rotation de Mandelbrot] Soit $t \geq 2$. L'ensemble M_t est **invariant par rotation** d'un angle de $\frac{2\pi}{t-1}$.

Démonstration : Posons $\forall n \in \mathbb{N}$, $H_n =$ "$\forall c \in \mathbb{C}, \forall t \geq 2,\ z_{e^{\frac{i2\pi}{t-1}}c_n} = e^{\frac{i2\pi}{t-1}} z_{c_n}$"

Initialisation : pour $n = 0$. Soit $c \in \mathbb{C}$ et $t \geq 2$. On a bien $0 = 0$.

Hérédité : Soit $n \in \mathbb{N}$ tel que H_n vraie. Soit $c \in \mathbb{C}$ et $t \geq 2$. On a :

$$\begin{aligned} z_{e^{\frac{i2\pi}{t-1}}c_{n+1}} &= \left(z_{e^{\frac{i2\pi}{t-1}}c_n}\right)^t + e^{\frac{i2\pi}{t-1}}c \\ &= \left(e^{\frac{i2\pi}{t-1}}z_{c_n}\right)^t + e^{\frac{i2\pi}{t-1}}c \text{ (par hypothèse de récurrence)} \\ &= e^{\frac{i2\pi}{t-1}}\left(e^{2i\pi}z_{c_n}^t + c\right) \\ &= e^{\frac{i2\pi}{t-1}}z_{c_{n+1}} \end{aligned}$$

En passant au module, on obtient donc que $z_{e^{\frac{i2\pi}{t-1}}c_n}$ et z_{c_n} ont même module, donc que l'un appartient à $\mathscr{M}$ si et seulement si l'autre y appartient. $\mathscr{M}$ est donc invariant par rotation d'un angle de $\frac{2\pi}{t-1}$.

Chapitre 5

Des fractales particulières : les courbes qui remplissent l'espace

5.1 Généralités

5.1.1 Introduction aux courbes qui remplissent l'espace

Quand on parle de fractales, on peut faire allusion à des constructions qui ont une forme hors du commun (comme *l'ensemble de Mandelbrot*), ou bien à des constructions qui ressemblent à des objets connus, comme un carré rempli par une courbe. En effet, ces courbes ont, de loin, une forme de carré, mais lorsque l'on tente de se rapprocher, et d'en observer un petit morceau, on remarque qu'elles ressemblent décidément beaucoup plus à une fractale qu'à une vulgaire courbe. Les mathématiciens nous ont légué beaucoup d'exemples de telles courbes, comme la *courbe de Peano* :

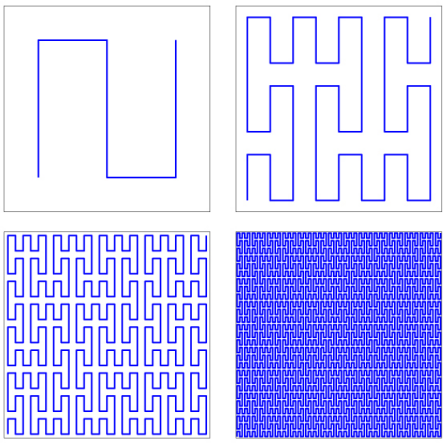


Figure 31. *La courbe de Peano, basée sur la répétition d'un motif à l'infini pour remplir le carré unité. Ses propriétés ne seront pas étudiées ici.*

On se propose ici d'étudier quelques aspects de ces courbes intrigantes, et d'en construire une. On se limitera ici aux courbes qui remplissent le carré unité, c'est à dire aux courbes surjectives de $[0,1]$ dans $[0,1]^2$.

5.1.2 Résultats préliminaires

On donne ici quelques résultats préliminaires qui seront utiles tout au long de cette partie

Résultat préliminaire 1 : Soit E un espace topologique. E est connexe si et seulement si toute fonction continue $g : E \longrightarrow \{0,1\}$ est constante

Démonstration : $\Rightarrow$ supposons que E soit connexe. Soit $g : E \longrightarrow \{0,1\}$ continue. On a $\{0\}$ et $\{1\}$ ouverts et disjoints, donc $f^{-1}(\{0\})$ et $f^{-1}(\{1\})$ sont ouverts et disjoints (la continuité de f conserve le caractère ouvert, et l'image réciproque de deux ensembles disjoints et encore disjointe), et leur réunion vaut E. Par connexité de E, on a que $f^{-1}(\{0\}) = \varnothing$ ou $f^{-1}(\{1\}) = \varnothing$, donc que f est constante égale à 1 ou à 0 respectivement.

$\Leftarrow$ on raisonne par contraposée. Supposons que E ne soit pas connexe, et montrons qu'il existe une fonction continue de E dans $\{0,1\}$ non constante. Comme E n'est pas connexe, il existe une partie à la fois ouverte et fermée de E, différente de E et de $\varnothing$. La fonction indicatrice de cette partie est continue dans $\{0,1\}$, mais n'est pas constante.

Résultat préliminaire 2 : Soit E et F deux espaces topologiques, et $f : E \longrightarrow F$ une fonction continue. Supposons que E soit connexe. Alors $f(E)$ est lui aussi connexe.

Démonstration : soit $g : f(E) \longrightarrow \{0,1\}$ continue. On a que $g \circ f : E \longrightarrow \{0,1\}$ est continue. Par connexité de E, le *résultat préliminaire 1* (sens direct) fournit que $g \circ f$ est constante. Donc en particulier, g est constante. Le *résultat préliminaire 1* (sens réciproque) fournit que $f(E)$ est connexe.

Définition : un **homéomorphisme** est une fonction bijective continue, dont la bijection réciproque est elle aussi continue.

Résultat préliminaire 3 : Soit E et F deux espaces topologiques et $f : E \longrightarrow F$. Si l'image réciproque par f d'un ouvert est encore un ouvert, alors f est continue.

Démonstration : soit $x \in E$ et $\varepsilon > 0$. L'image réciproque de la boule ouverte $B_F(f(x), \varepsilon)$ est un ouvert de E qui contient x. Donc fixons $\delta > 0$ tel que la boule ouverte $B_E(x, \delta) \subset f^{-1}(B_F(f(x), \varepsilon)$. Alors en passant à l'image directe, on a que $f(B_E(x, \delta) \subset B_F(f(x), \varepsilon)$, donc que f est continue en x. Donc f est continue.

Résultat préliminaire 4 : Soit E et F deux espaces topologiques et $f : E \longrightarrow F$. Si l'image réciproque par f d'un fermé est encore un fermé, alors f est continue.

Démonstration : Soit A un ouvert de F. Soit B le complémentaire de A dans F, fermé donc. On a $f^{-1}(A) = f^{-1}(F\backslash B) = f^{-1}(F)\backslash f^{-1}(B) = E\backslash f^{-1}(B)$. Or, $f^{-1}(B)$ est fermé par hypothèse, donc $E\backslash f^{-1}(B)$ est ouvert. Donc l'image réciproque de f envoie les ouverts sur les ouverts. On conclut par le *résultat préliminaire 3*.

Résultat préliminaire 5 : Soit E, F des espaces topologiques, et $f : E \longrightarrow F$ une fonction continue. Soit $A \subset E$ compact. Alors $f(A)$ est compact.

Démonstration : Soit $\mathscr{U}$ un recouvrement ouvert de $f(A)$. Comme f est continue, $\mathscr{V} = \left\{f^{-1}(U), U \in \mathscr{U}\right\}$ est un recouvrement ouvert de A. Comme A est compact, $\mathscr{V}$ admet un sous-recouvrement fini $\mathscr{V}'$. Puis $\{f(V), V \in \mathscr{V}'\}$ est un sous-recouvrement fini de $\mathscr{U}$. Donc $f(A)$ est compact.

Résultat préliminaire 6 : Soit E et F deux espaces topologiques, et $f : E \longrightarrow F$ bijective et continue. Supposons que E soit compact et F séparé. Alors f est un **homéomorphisme**.

Démonstration : soit X un fermé de E. X est un compact, donc $f(X)$ est compact par le *résultat préliminaire 5*. Par séparation de F, $f(X)$ est fermé dans F. Ainsi, f envoie des fermés sur des fermés. En voyant f comme la fonction réciproque de f^{-1}, on a par le *résultat préliminaire 4* que f^{-1} est continue. Donc f est un homéomorphisme.

Résultat préliminaire 7 : Soit (u_n) et (v_n) deux suites réelles à valeurs positives. Si $\sum_{n=1}^{+\infty} u_n$ existe, et que v_n est majorée, alors $\sum_{n=1}^{+\infty} u_n v_n$ existe

Démonstration : Posons $S = \sum_{n=1}^{+\infty} u_n$.

Déjà, $\forall N \in \mathbb{N}^*,\ \sum_{n=1}^{N+1} u_n - \sum_{n=1}^{N} u_n = u_{N+1} \geq 0$, donc $\left(\sum_{n=1}^{N} u_n\right)_{N\geq 1}$ est croissante. De plus elle tend vers S, donc elle est *majorée* par S. Puis, soit M un majorant de (v_n). Alors $\forall n \in \mathbb{N},\ u_n v_n \leq u_n M$, donc il vient :

$$\forall N \in \mathbb{N}^*,\ \sum_{n=1}^{N} u_n v_n \leq \sum_{n=1}^{N} u_n M$$

$$\text{donc } \forall N \in \mathbb{N}^*,\ \sum_{n=1}^{N} u_n v_n \leq M \sum_{n=1}^{N} u_n$$

$$\text{donc } \forall N \in \mathbb{N}^*,\ \sum_{n=1}^{N} u_n v_n \leq MS$$

donc $\left(\sum_{n=1}^{N} u_n v_n\right)_{N\geq 1}$ est *majorée*. De plus, elle est *croissante* (on le montre de la même manière que la somme des termes de (u_n) l'est), donc le *théorème de la limite monotone* montre qu'elle admet une *limite finie*, donc $\sum_{n=1}^{+\infty} u_n v_n$ existe.

5.2 De la continuité et de la dérivabilité de telles courbes

Créer une courbe qui remplit l'espace peut paraître contre-intuitif au premier abord, et d'autant plus lorsque l'on exige que celle-ci soit continue. Mais c'est néanmoins possible, comme nous le démontrerons lors de la construction de la courbe, un peu plus loin. Cependant, une telle courbe ne pourra être bijective, comme le montre le :

Théorème : il n'existe pas de fonctions continues et bijectives de $[0,1]$ dans $[0,1]^2$

Démonstration : on raisonne par l'absurde. Soit $f : [0,1] \longrightarrow [0,1]^2$ continue et bijective. f étant continue sur le compact $[0,1]$, on a que f^{-1} est elle aussi continue par le *résultat préliminaire 6*.

Soit $c \in]0,1[$. On a $[0,1]^2 \backslash \{f(c)\}$ qui est connexe par arcs, donc connexe. Comme f^{-1} est continue, $f^{-1}([0,1]^2 \backslash \{f(c)\})$ est lui aussi connexe (par le *résultat préliminaire 2*).

Le caractère bijectif de f fournit $f^{-1}([0,1]^2 \backslash \{f(c)\}) = f^{-1}([0,1]^2) \backslash \{c\} = [0,1] \backslash \{c\} = [0,c[\cup]c,1]$, qui n'est pas connexe. D'où contradiction.

On peut demander qu'une telle courbe soit dérivable également. Mais on se heurte cette fois ci au résultat suivant :

Théorème : il n'existe pas de fonction surjective et de classe $\mathscr{C}^1$ de $[0,1]$ dans $[0,1]^2$.

Démonstration : On raisonne par l'absurde. Supposons qu'il existe une telle fonction f. L'idée est ici de recouvrir $f([0,1])$ par une surface d'aire inférieure à celle de $[0,1]^2$, ce qui sera absurde.

Soit $||.||$ la norme suivante sur $\mathbb{R}^2$: $||(x,y)|| = sup\{|x|,|y|\}$. Posons $M = \underset{t\in[0,1]}{sup} ||f'(t)||$. Soit $n \in \mathbb{N}^*$, et σ la subdivision de $[0,1]$: $0 < \frac{1}{n} < \ldots \frac{n-1}{n} < 1$.

Soit $k \in [\![0, n-1]\!]$. L'inégalité des accroissements finis appliquée à f fournit :

$$\forall t \in \left[\frac{k}{n},\ \frac{k+1}{n}\right], \left|\left|f(t) - f\left(\frac{k+1/2}{n}\right)\right|\right| \leq M \left|t - \frac{k+1/2}{n}\right| \leq \frac{M}{2n}$$

De cette manière, on interprète graphiquement que $f\left(\left[\frac{k}{n}, \frac{k+1}{n}\right]\right)$ est inclus dans le carré de centre $f\left(\frac{k+1/2}{n}\right)$ et de côté $\frac{M}{n}$. On le notera $C_{n,k}$. On en déduit que :

$$\begin{aligned} f([0,1]) &= f\left(\bigcup_{k=0}^{n-1}\left[\frac{k}{n}, \frac{k+1}{n}\right]\right) \\ &= \bigcup_{k=0}^{n-1} f\left(\left[\frac{k}{n}, \frac{k+1}{n}\right]\right) \\ &\subset \bigcup_{k=0}^{n-1} C_{n,k} =: S_n \end{aligned}$$

Comme les $C_{n,k}$ ne sont pas nécessairement deux à deux disjoints, on a que l'aire (notée $\mathscr{A}(.)$) de S_n vérifie :

$$\begin{aligned} \mathscr{A}(S_n) &\leq \sum_{k=0}^{n-1} \mathscr{A}\left(C_{n,k}\right) \\ &= \sum_{k=0}^{n-1} \left(\frac{M}{n}\right)^2 \\ &= \frac{M^2}{n} \end{aligned}$$

Fixons enfin $n_0 \in \mathbb{N}^*$ tel que $n_0 > M$. On a $\dfrac{M^2}{n} < 1$, et par l'inégalité ci-dessus, $\mathscr{A}(S_n) < 1$. Comme $\mathscr{A}([0,1]^2) = 1$, on a nécessairement $[0,1]^2 \not\subset S_n$. Or, $f([0,1]) \subset S_n$, donc $[0,1]^2 \not\subset f([0,1])$; absurde, car $f([0,1]) = [0,1]^2$ par définition, ce qui achève la démonstration.

Ainsi, ces courbes continues, bien qu'elles existent, ne sont pas injectives, et leurs dérivées ne sont pas continues.

5.3 Construction d'une courbe continue

Nous allons construire ici une courbe continue remplissant le carré unité, de manière théorique en premier lieu, puis grâce à l'outil informatique.

5.3.1 Principe de fonctionnement

L'astuce pour construire une courbe remplissant l'espace est de non pas créer une fonction *classique* (dans le sens où elle associe à chaque abscisse une ordonnée), mais au contraire de créer deux fonctions, une codant pour l'abscisse et une autre pour l'ordonnée, toutes deux en fonction d'un paramètre t, que l'on

peut qualifier de *temporel*.

L'image qu'il faut donc se faire de la courbe est celle d'un mobile se déplaçant sur le carré unité sur un intervalle de temps de 1 seconde, et qui, au bout de ce laps, aura parcouru l'intégralité de la surface, de façon *continue*... Plus rigoureusement, il s'agit donc de créer deux fonction x et y, chacune continue et surjective, de la forme :

$$\begin{array}{rccc} x & : & [0,1] & \to & [0,1] \\ & & t & \mapsto & x(t) \end{array}$$

$$\begin{array}{rccc} y & : & [0,1] & \to & [0,1] \\ & & t & \mapsto & y(t) \end{array}$$

5.3.2 Construction d'une courbe continue remplissant le carré unité

Définition : Pour cet exemple, commençons par poser deux fonctions f et g sur $\mathbb{R}$, que l'on définit d'abord sur $[0,1[$ ainsi, et que l'on étend ensuite par 1-périodicité :

$$f(t) = \begin{cases} 0 & \text{si} \quad 0 \leq t < 0.4 \\ 10t - 4 & \text{si} \quad 0.4 \leq t < 0.5 \\ 1 & \text{si} \quad 0.5 \leq t < 0.8 \\ -10t + 9 & \text{si} \quad 0.8 \leq t < 0.9 \\ 0 & \text{si} \quad 0.9 \leq t < 1 \end{cases}$$

$$g(t) = \begin{cases} 0 & \text{si} \quad 0 \leq t < 0.2 \\ 10t - 2 & \text{si} \quad 0.2 \leq t < 0.3 \\ 1 & \text{si} \quad 0.3 \leq t < 0.4 \\ -10t + 5 & \text{si} \quad 0.4 \leq t < 0.5 \\ 0 & \text{si} \quad 0.5 \leq t < 0.6 \\ 10t - 6 & \text{si} \quad 0.6 \leq t < 0.7 \\ 1 & \text{si} \quad 0.7 \leq t < 0.8 \\ -10t + 9 & \text{si} \quad 0.8 \leq t < 0.9 \\ 0 & \text{si} \quad 0.9 \leq t < 1 \end{cases}$$

Proposition : f et g sont continues sur $\mathbb{R}$

Démonstration : Démontrons-le pour f (la preuve est symétrique pour g). Déjà, on a $\lim\limits_{t \to 0^+} f(t) = 0$, par définition de f sur $[0,1[$, et $\lim\limits_{t \to 0^-} f(t) = 0$, par extension par 1-périodicité à $[-1,0[$. On obtient les mêmes résultats en 1. Il suffit donc de montrer que f est continue sur $]0,1[$ et on étendra à $\mathbb{R}$ par 1-périodicité. Soit $t \in]0,1[$.

Supposons que $t \notin \{0.4, 0.5, 0.8, 0.9\}$. Alors f coïncide au voisinage de t avec une fonction continue (soit constante, soit linéaire, par définition de f), donc f est continue en t.

Supposons que $t \in \{0.4, 0.5, 0.8, 0.9\}$. Étudions le cas $t = 0.4$, et les autres cas se déduiront de la même manière. On a $\lim_{t \to 0.4^-} f(t) = 0$ (par définition de f sur $[0, 0.4[$) et $\lim_{t \to 0.4^+} f(t) = 0$ (car $t \mapsto 10t - 4$ tend vers 0 lorsque t tend vers 0.4), ce qui achève la preuve.

Définition : On définit x et y (abscisse et ordonnée de la courbe) de la manière suivante :

$$x(t) = \sum_{n=1}^{+\infty} \frac{f(10^{n-1}t)}{2^n} \text{ et } y(t) = \sum_{n=1}^{+\infty} \frac{g(10^{n-1}t)}{2^n}$$

Proposition : x et y sont bien définies

Démonstration : Démontrons-le pour x (la preuve est identique pour y).
Déjà, $\sum_{n=1}^{+\infty} \frac{1}{2^n}$ existe, en tant que série ayant pour terme général une suite géométrique de raison $\frac{1}{2}$, qui vérifie $\left|\frac{1}{2}\right| < 1$. De plus, $\left(\frac{1}{2^n}\right)_{n \geq 1}$ est à valeurs positives, tout comme f. Enfin, f est majorée (par 1 par exemple), donc le *résultat préliminaire 1* fournit l'existence de x.

Proposition : x et y sont continues

Démonstration : Démontrons-le pour x (la preuve est identique pour y).
Soit $N \in \mathbb{N}$ et posons $\forall t \in \mathbb{R}, \theta(t) = \sum_{n=1}^{N} \frac{f(10^{n-1}t)}{2^n}$

Soit $p \in \mathbb{N}^*,\ t \in \mathbb{R}$. On a déjà :

$$\sum_{n=1}^{N+p} \frac{f(10^{n-1}t)}{2^n} - \theta(t) = \sum_{n=N+1}^{N+p} \frac{f(10^{n-1}t)}{2^n}. \tag{5.1}$$

Or,

$$\sum_{n=1}^{N+p} \frac{f(10^{n-1}t)}{2^n} - \theta(t) \underset{N \to +\infty}{\longrightarrow} x(t) - x(t) = 0$$

Donc

$$\sum_{n=N+1}^{N+p} \frac{f(10^{n-1}t)}{2^n} \underset{N \to +\infty}{\longrightarrow} 0$$

.

Soit $\varepsilon > 0$. On peut donc fixer $N \in \mathbb{N}$, tel que

$$\forall n \in \mathbb{N},\ n \geq N \Rightarrow \left| \sum_{n=N+1}^{N+p} \frac{f(10^{n-1}t)}{2^n} \right| \leq \varepsilon$$

.

Ensuite, en faisant tendre p vers $+\infty$ dans (5.1), on obtient :

$$|x(t) - \theta(t)| \leq \varepsilon$$

Puis θ est continue (ce caractère est issu de la continuité de f et des opérations algébriques usuelles), donc fixons $\delta > 0$, tel que :

$$\forall u \in [t-\delta, t+\delta],\ |\theta(t) - \theta(u)| \leq \varepsilon$$

Enfin, montrons que x est continue en t. Soit $u \in [t-\delta, t+\delta]$. On a :

$$|x(t) - x(u)| = |x(t) + \theta(t) - \theta(t) - x(u) + \theta(u) - \theta(u)|$$

Donc l'inégalité triangulaire donne :

$$|x(t) - x(u)| \leq |x(t) - \theta(t)| + |\theta(t) - \theta(u)| + |\theta(u) - x(u)|$$

Donc :

$$|x(t) - x(u)| \leq 3\varepsilon$$

Donc x est continue en t, ce qui achève la preuve.

Proposition :

$$\forall t \in \mathbb{R},\ (x(t), y(t)) \in [0,1]^2$$

Démonstration : Démontrons-le pour x (démonstration identique pour y). Soit $t \in \mathbb{R}$. Déjà, on a $\forall n \in \mathbb{N},\ 0 \leq f(10^{n-1}t) \leq 1$ ainsi que $\forall n \in \mathbb{N},\ \dfrac{1}{2^n} \geq 0$. Cela permet de déduire :

- D'une part, $\forall n \in \mathbb{N},\ \dfrac{f(10^{n-1}t)}{2^n} \geq 0$, donc $\forall N \in \mathbb{N},\ \displaystyle\sum_{n=1}^{N} \frac{f(10^{n-1}t)}{2^n} \geq 0$, donc en passant à la limite, $x(t) \geq 0$.
- D'autre part, $\forall n \in \mathbb{N},\ \dfrac{f(10^{n-1}t)}{2^n} \leq \dfrac{1}{2^n}$, donc

$$\forall N \in \mathbb{N},\ \sum_{n=1}^{N} \frac{f(10^{n-1}t)}{2^n} \leq \sum_{n=1}^{N} \frac{1}{2^n}. \tag{5.2}$$

  Or, soit $N \in \mathbb{N}$. On a

$$\sum_{n=1}^{N} \frac{1}{2^n} = \frac{1}{2} \cdot \frac{1 - \frac{1}{2^N}}{1 - \frac{1}{2}}.$$

  Donc en faisant tendre N vers $+\infty$, on obtient

$$\sum_{n=1}^{+\infty} \frac{1}{2^n} = \frac{1}{2} \cdot 2 = 1.$$

  Ainsi, (5.2) fournit $x(t) \leq 1$.

Définition : On pose la fonction ϕ suivante :

$$\begin{array}{rccc} \phi & : & [0,1] & \to & [0,1]^2 \\ & & t & \mapsto & (x(t), y(t)) \end{array}$$

Proposition : ϕ est continue

Démonstration : elle l'est par héritage de la continuité de x et y

Proposition : ϕ est surjective

Démonstration : Soit $(\alpha, \beta) \in [0,1]^2$, et montrons que

$$\exists t \in [0,1],\ \phi(t) = (\alpha, \beta).$$

On va utiliser la décomposition des réels : fixons (α_n) et (β_n) des suites à valeurs dans $\{0,1\}$, telles que $\alpha = \sum_{n=1}^{+\infty} \frac{\alpha_n}{2^n}$ et $\beta = \sum_{n=1}^{+\infty} \frac{\beta_n}{2^n}$ (décomposition en base 2). Construisons ensuite une suite t_n ainsi :

$$\left\{ \begin{array}{lll} t_n = 1 & \text{si} & (\alpha_n, \beta_n) = (0,0) \\ t_n = 3 & \text{si} & (\alpha_n, \beta_n) = (0,1) \\ t_n = 5 & \text{si} & (\alpha_n, \beta_n) = (1,0) \\ t_n = 7 & \text{si} & (\alpha_n, \beta_n) = (1,1) \end{array} \right.$$

et on pose $t = \sum_{n=1}^{+\infty} \frac{t_n}{10^n}$ (décomposition en base 10). Montrons que t ainsi défini convient.

Soit $n \in \mathbb{N}^*$. On pose $\mu_n = \sum_{k=1}^{n-1} t_k 10^{n-1-k} \in \mathbb{N}$ et $\rho_n = 10^{n-1}t - \mu_n - \frac{t_n}{10}$. On a alors :

$$10^{n-1}t = \mu_n + \frac{t_n}{10} + \rho_n.$$

Puis montrons que $\rho_n \in [0, 0.1]$. Soit $N \geq n$. On a :

$$\begin{array}{rcl} 10^{n-1}\left(\sum_{k=1}^{N} \frac{t_k}{10^k}\right) - \mu_n - \frac{t_n}{10} & = & \sum_{k=1}^{N} t_k 10^{n-1-k} - \mu_n - \frac{t_n}{10} \\ & = & \sum_{k=n}^{N} t_k 10^{n-1-k} - \frac{t_n}{10} \\ & = & \sum_{k=n+1}^{N} t_k 10^{n-1-k} \\ & = & \sum_{k=1}^{N-n} t_{k+n} 10^{-1-k} \\ & = & \frac{1}{10} \sum_{k=1}^{N-n} \frac{t_{k+n}}{10^k} \end{array}$$

- D'une part, $\dfrac{1}{10}\displaystyle\sum_{k=1}^{N-n}\frac{t_{k+n}}{10^k} \geq 0$.
- D'autre part, $\forall k \in \mathbb{N},\ t_k \leq 9$, donc

$$\begin{aligned}\frac{1}{10}\sum_{k=1}^{N-n}\frac{t_{k+n}}{10^k} &\leq \frac{9}{10}\sum_{k=1}^{N-n}\frac{1}{10^k}\\ &= \frac{9}{10}\cdot\frac{1}{10}\cdot\frac{1-(1/10)^{N-n}}{1-1/10}\\ &\leq \frac{9}{10}\cdot\frac{1}{10}\cdot\frac{1}{9/10}\\ &= \frac{1}{10}\end{aligned}$$

Donc

$$\left(10^{n-1}\left(\sum_{k=1}^{N}\frac{t_k}{10^k}\right)-\mu_n-\frac{t_n}{10}\right)\in[0,0.1]\,.$$

En faisant tendre N vers $+\infty$, on obtient $\rho_n \in [0, 0.1]$.

Soit $N \in \mathbb{N}^*$. On a désormais :

$$\begin{aligned}\sum_{n=1}^{N}\frac{f(10^{n-1})}{2^n} &= \sum_{n=1}^{N}\frac{f\left(\mu_n+\frac{t_n}{10}+\rho_n\right)}{2^n}\\ &= \sum_{n=1}^{N}\frac{f\left(\frac{t_n}{10}+\rho_n\right)}{2^n}\ \ (f \text{ 1-périodique})\end{aligned}$$

Puis, par définition de f et par le fait que $0 \leq \rho_n \leq \dfrac{1}{10}$, on obtient :

$$\begin{cases}\text{si } t_n = 1, \text{ alors } f\left(\dfrac{t_n}{10}+\rho_n\right) = f\left(\dfrac{1}{10}\right) = 0 = \alpha_n\\ \text{si } t_n = 3, \text{ alors } f\left(\dfrac{t_n}{10}+\rho_n\right) = f\left(\dfrac{3}{10}\right) = 0 = \alpha_n\\ \text{si } t_n = 5, \text{ alors } f\left(\dfrac{t_n}{10}+\rho_n\right) = f\left(\dfrac{5}{10}\right) = 1 = \alpha_n\\ \text{si } t_n = 7, \text{ alors } f\left(\dfrac{t_n}{10}+\rho_n\right) = f\left(\dfrac{7}{10}\right) = 1 = \alpha_n\end{cases}$$

On a donc $\displaystyle\sum_{n=1}^{N}\frac{f\left(10^{n-1}t\right)}{2^n} = \sum_{n=1}^{N}\frac{\alpha_n}{2^n}$, donc l'unicité de la limite (en faisant tendre N vers $+\infty$) donne $x(t) = \alpha$. On utilise un raisonnement symétrique pour montrer que $y(t) = \beta$ (en passant par la fonction g). Donc t convient, donc ϕ et surjective, et la preuve est achevée.

5.3.3 Visualisation de la courbe

Maintenant que nous avons montré que cette courbe remplit bien le carré unité de façon continue, il est temps de la visualiser. On se propose de le faire à

l'aide d'un programme Python. Cependant, il existe un obstacle qu'on ne peut pas contourner, celui des *sommes infinies*. Pour remédier à ce problème, on va majorer l'erreur commise, et faire en sorte qu'elle ne dépasse pas la largeur d'un pixel, afin qu'elle ne perturbe pas l'affichage de la courbe, et qu'elle soit fidèle à son modèle théorique.

Définition : On définit *l'erreur d'ordre* N de x en t comme étant la somme suivante (définition symétrique pour y) :

$$E_x(t,N) = \sum_{n=N+1}^{+\infty} \frac{f(10^{n-1}t)}{2^n}$$

On remarque que

$$E_x(t,N) = x(t) - \sum_{n=1}^{N} \frac{f(10^{n-1}t)}{2^n}.$$

Proposition : *L'erreur* vérifie :

$$\forall N \in \mathbb{N},\ \forall t \in [0,1],\ E_x(t,N) \leq \frac{1}{2^N}$$

Démonstration : Soit $t \in \mathbb{R}$. On a $\forall n \in \mathbb{N}^*,\ f(10^{n-1}t) \leq 1$, donc

$$\forall n \in \mathbb{N}^*,\ \frac{f(10^{n-1}t)}{2^n} \leq \frac{1}{2^n}.$$

Comme les séries ayant pour terme général ces deux suites convergent, et qu'elles sont à valeurs positives, il vient :

$$\forall N \in \mathbb{N},\ \sum_{n=N+1}^{+\infty} \frac{f(10^{n-1}t)}{2^n} \leq \sum_{n=N+1}^{+\infty} \frac{1}{2^n}$$

Donc :

$$\forall N \in \mathbb{N},\ E_x(t,N) \leq \lim_{p\to+\infty} \left(\frac{1}{2^{N+1}} \cdot \frac{1-(1/2)^{p-(N+1)}}{1-1/2} \right)$$

Donc :

$$\forall N \in \mathbb{N},\ E_x(t,N) \leq \frac{1}{2^{N+1}} \cdot 2$$

Donc :

$$\forall N \in \mathbb{N},\ E_x(t,N) \leq \frac{1}{2^N}$$

Pour un écran à haute résolution, la largeur est de 1080 pixels. Étant donné que le carré a une longueur 1, on en déduit que chaque pixel représente $\frac{1}{1080}$ d'unité. On remarque que $N = 11$ est le premier ordre permettant que l'erreur commise ne soit pas visible, car elle vaut alors $\frac{1}{2048}$. Les *sommes infinies* deviennent donc des sommes indexées jusqu'à 11 en Python.

Voici ci-après des graphes de courbes obtenus par simulation informatique (le programme *Python* correspondant se trouve en *annexe 4*), avec des pas de temps variables :

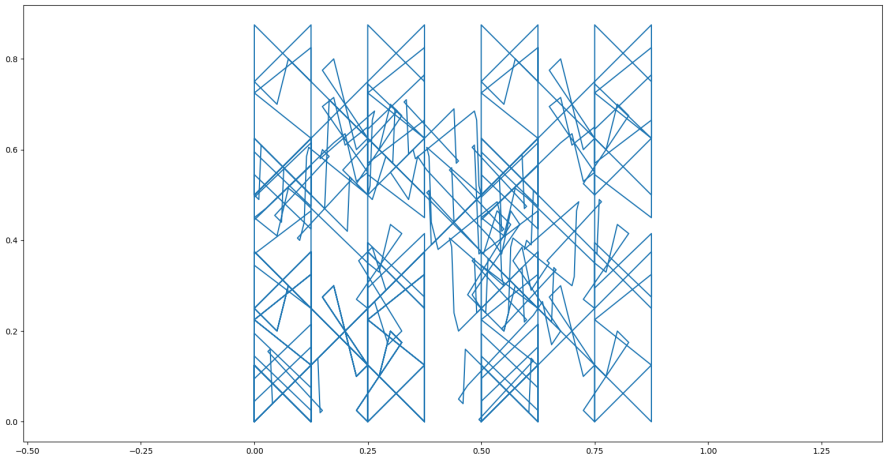


Figure 32. *Affichage avec un pas de* 0,001.

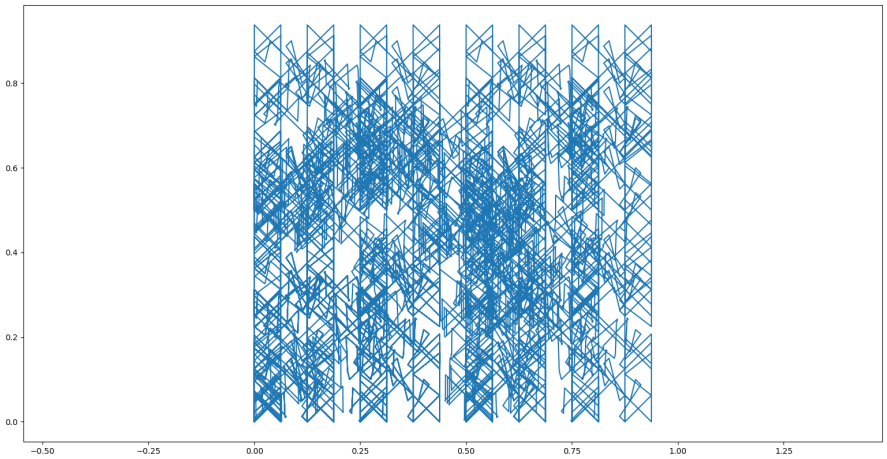


Figure 33. *Affichage avec un pas de* 0,0001.

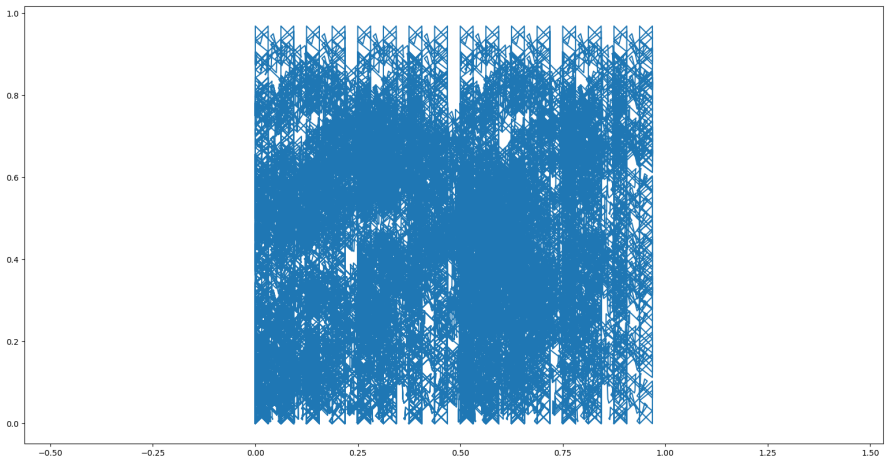


Figure 34. *Affichage avec un pas de* 0,00001.

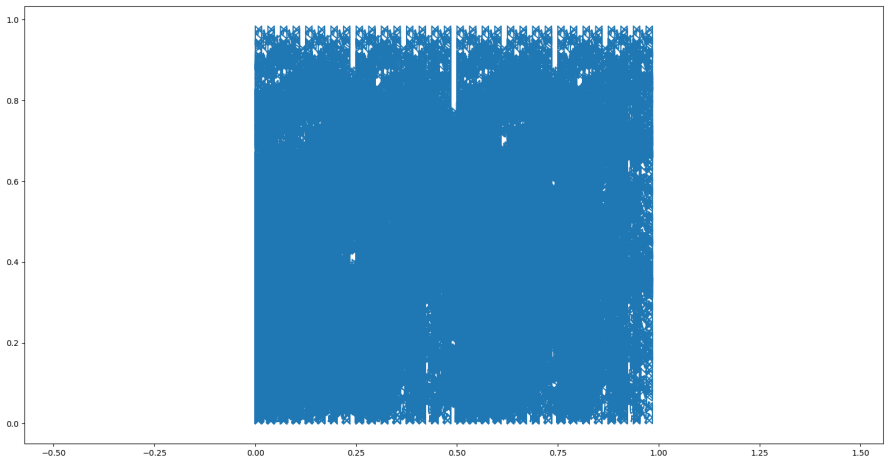


Figure 35. *Affichage avec un pas de* 0,000001.

Commentaires : On remarque immédiatement qu'il faut un pas très faible pour commencer à observer des résultats satisfaisants, avec peu de zones blanches visibles. On observe cependant qu'une grande bande blanche centrale est recouverte assez lentement, mais on peut, de manière raisonnable, se permettre de supposer qu'avec un pas beaucoup plus faible (non calculable par *Python*), le carré paraîtra totalement rempli.

On remarque aussi que les zones extrêmes (*ie* d'abscisses et d'ordonnées proches de 1) se remplissent d'autant plus que le pas est faible, comme si la courbe "gonflait" vers le haut et vers la droite.

Chapitre 6

Annexes

Dans ces annexes sont présentées les programmes Pythons utilisés dans les chapitres précédents.

6.1 Annexe 1 : programme Python pour afficher le flocon de Koch

Ci-après le programme permettant d'afficher le **flocon de Koch**.

```
import numpy as np
import matplotlib.pyplot as plt

#flocon([0,1,1/2,0],[0,0,-np.sqrt(3)/2,0],8)
#flocon([0,1],[0,0],8)

#animation(9,1.01,0.03,400)

"""
En entrée : les 2 points à partir desquels on veut créer
    le "3ème sommet du triangle" par rotation de pi/3
En sortie : le point voulu
"""
def rotation(x1,y1,x2,y2):

    z=(0.5+1j*(np.sqrt(3)/2))*(x2+1j*y2-(x1+1j*y1))+x1+1j
        *y1

    return np.real(z),np.imag(z)


"""
En entrée : la liste codant pour le flocon d'ordre de pré
    cision n
```

```
Le but de cette fonction est exactement le principe du
    flocon de Koch : créer les petites "pointes" entre 2
    points déjà construits.
En sortie : la liste codant pour le flocon d'ordre de pré
    cision n+1
"""
def division(abscisse,ordonnee):

    newx=[abscisse[0]]
    newy=[ordonnee[0]]

    for t in range(1,len(abscisse)):

     #Ici, le prinipe est de fixer nos 2 points desquels
         on créera les 3 autres : 2 repectivement au tiers
          et aux deux-tiers du segment, et un 3ème tel que
          les 3 nouveaux points créés forment un triangle
         équilatéral

        a1=abscisse[t-1]
        o1=ordonnee[t-1]
        a2=abscisse[t]
        o2=ordonnee[t]

        abstiers=(2*a1+a2)/3
        absdeuxtiers=(a1+2*a2)/3
        ordtiers=(2*o1+o2)/3
        orddeuxtiers=(o1+2*o2)/3
        #Ce points représentent les 2 premiers points du
            triangle équilatéral

        nvpt=rotation(abstiers,ordtiers,absdeuxtiers,
            orddeuxtiers)
        #Ce point représente le 3ème point du triangle é
            quilatéral

        newx.append(abstiers)
        newx.append(nvpt[0])
        newx.append(absdeuxtiers)
        newx.append(a2)

        newy.append(ordtiers)
        newy.append(nvpt[1])
        newy.append(orddeuxtiers)
        newy.append(o2)

    return newx,newy


"""
```

```
Une fois les fonctions auxiliaires construites, la
    fonction qui retourne les abscisses et ordonnées des
    points de passage du flocon de Koch à la précision n
    voulue est relativement simple : il suffit juste d'
    appliquer n fois la fonction d'augmentation de pré
    cision de notre flocon.

NB : on laisse le libre choix des 2 coordonnées des 2
    premiers points à partir desquels on construit notre
    flocon : les listes x et y en entrée représentent ce
    degré de lierté. Généralement, on utilise les listes
    [0,1] et [0,0] (liste des abscisses et lste des ordonn
    ées) codant pour les points de coordonnées (0,0) et
    (1,0) afin de conserver une cohérence avec la
    construction théorique du flocon de Koch.
"""
def flocon(x,y,n):

    if n==0:
        plt.plot(x,y)
        plt.axis("equal")
        plt.show()

    else:
        tableau=division(x,y)
        flocon(tableau[0],tableau[1],n-1)

"""
Cet algorithme a la même fonction que celui ci-haut, à la
     différence qu'il retourne la liste des abscisses et
    des ordonnées au lieu de la figure, ce qui nous
    servira à notre animation.
"""
def floconpouranimation(n):

    x=[0,1]
    y=[0,0]
    m=n

    while m>0:

        tableau=division(x,y)
        x,y=tableau[0],tableau[1]
        m-=1

    return x,y


"""
```

```
Cet algorithme utilise le principe de l'effet de bord
    pour donner l'illusion d'un zoom sur le flocon.
Il se base en fait sur l'agrandissement progressif du
    flocon par dilatation de coordonnées.
"""
def animation(precision,zoom,vitesse_zoom,iterations):

    x=floconpouranimation(precision)[0]
    y=floconpouranimation(precision)[1]

    graphe=plt.plot(x,y)[0]
    plt.axis("equal")

    for k in range(iterations):

        for i in range(0,len(x)):
            x[i]*=zoom
            y[i]*=zoom

        plt.pause(vitesse_zoom)

        graphe.set_xdata(x)
        graphe.set_ydata(y)

    plt.show()
```

6.2 Annexe 2 : programme Python pour calculer la dimension du flocon de Koch

Ci-après le programme permettant de calculer la dimension du **flocon de Koch**.

```
import numpy as np
import matplotlib.pyplot as plt



### Partie 1 : construction des coordonnées des points de
    passsage du flocon de Koch
"""
En entrée : les 2 points à partir desquels on veut créer
    le "3ème sommet du triangle" par rotation de pi/3
En sortie : le point voulu
"""
def rotation(x1,y1,x2,y2):

    z=(0.5+1j*(np.sqrt(3)/2))*(x2+1j*y2-(x1+1j*y1))+x1+1j
        *y1
```

```
    return np.real(z),np.imag(z)




"""
En entrée : la liste codant pour le flocon d'ordre de pré
    cision n
Le but de cette fonction est exactement le principe du
    flocon de Koch : créer les petites "pointes" entre 2
    points déjà construits.
En sortie : la liste codant pour le flocon d'ordre de pré
    cision n+1
"""
def division(abscisse,ordonnee):

    newx=[abscisse[0]]
    newy=[ordonnee[0]]

    for t in range(1,len(abscisse)):

    #Ici, le prinipe est de fixer nos 2 points desquels
        on créera les 3 autres : 2 repectivement au tiers
        et aux deux-tiers du segment, et un 3ème tel que
        les 3 nouveaux points créés forment un triangle é
        quilatéral

        a1=abscisse[t-1]
        o1=ordonnee[t-1]
        a2=abscisse[t]
        o2=ordonnee[t]

        abstiers=(2*a1+a2)/3
        absdeuxtiers=(a1+2*a2)/3
        ordtiers=(2*o1+o2)/3
        orddeuxtiers=(o1+2*o2)/3
        #Ce points représentent les 2 premiers points du
            triangle équilatéral

        nvpt=rotation(abstiers,ordtiers,absdeuxtiers,
            orddeuxtiers)
        #Ce point représente le 3ème point du triangle é
            quilatéral

        newx.append(abstiers)
        newx.append(nvpt[0])
        newx.append(absdeuxtiers)
        newx.append(a2)
```

```
        newy.append(ordtiers)
        newy.append(nvpt[1])
        newy.append(orddeuxtiers)
        newy.append(o2)

    return newx,newy



"""
Une fois les fonctions auxiliaires construites, la
    fonction qui retourne les abscisses et ordonnées des
    points de passage du flocon de Koch à la précision n
    voulue est relativement simple : il suffit juste d'
    appliquer n fois la fonction d'augmentation de pré
    cision de notre flocon.

NB : on laisse le libre choix des 2 coordonnées des 2
    premiers points à partir desquels on construit notre
    flocon : les listes x et y en entrée représentent ce
    degré de lierté. Généralement, on utilise les listes
    [0,1] et [0,0] (liste des abscisses et lste des ordonn
    ées) codant pour les points de coordonnées (0,0) et
    (1,0) afin de conserver une cohérence avec la
    construction théorique du flocon de Koch.
"""
def flocon(x,y,n):

    while n>0:
        x,y=division(x,y)[0],division(x,y)[1]
        n-=1

    return x,y



### Partie 2 : algorithme de calcul de dimension
"""
Pour des raisons pratiques (précision du résultat, adé
    quation à la limite numérique des arguments pouvant ê
    tre pris par chacune des fonctions ultérieures), on
    considèrera le flocon d'ordre de précision 8 et de
    points de départ (0,0) et (1,0), c'est-à-dire le
    flocon codé par ([0,1],[0,0],8)
"""
```

```
"""
En entrée : les listes relatives au flocon de taille n
En sortie : la même liste mais agrémentée de nouveaux
    points de passages ne modifiant pas l'allure du flocon
Utilité de la fonction : elle est nécessaire au principe
    que l'on utilisera pour calculer la dimension du
    flocon de Koch.


En effet, la méthode qui sera utilisée ici repose sur un
    principe assez simplifié mais efficace des
    recouvrements finis qui définissent la dimension d'un
    espace métrique : on quadrille le plan par des
    carreaux de côté fixé (on verra plus tard que nous
    avons choisi des carreaux de taille 0.1 ici par
    commodité) puis on compte le nombre de carreaux touché
    s par le flocon.
On réitère le processus avec un quadrillage plus fin (qu'
    on émulera ici par un flocon plus grand par souci d'
    efficacité de l'alogrithme) puis on garde en mémoire
    les facteurs d'agrandissement du flocon ainsi que leur
     nombre de carreaux touchés relatifs.
La l'explication de la fin du procédé de calcul de
    dimension viendra ci-dessous.

D'où l'utilité de créer des nouveaux "points de passage",
     car il ne faut pas oublier que notre flocon est codé
    par 2 listes de coordonnées et qu'il est beaucoup plus
     difficile de savoir si une courbe traverse un carreau
     que de savoir si un point appartient à ce carreau.
"""
def CreationPtsIntermediaires(x,y,n):

    table=flocon(x,y,n)

    abscisses,ordonnees=table[0],table[1]
    m=25

    while len(abscisses)<100000 and m>0:
    #C'est un critère arbitraire mais qui permet d'avoir
        suffisament de points

        nvabscisses=[abscisses[0]]
        nvordonnees=[ordonnees[0]]
        l=len(abscisses)

        for k in range(l-1):
            if ((abscisses[k]-abscisses[k+1])**2+(
                ordonnees[k]-ordonnees[k+1])**2)
```

```
                **0.5>=0.0000001:
            #On maximise les chances que ce soit entre
                deux points plutôt éloignés quon crée de
                       nouveaux points de passage, ce qui
                est beaucoup plus utile. Le critère d'é
                loignement très large permet d'être sûr
                que de nouveaux points seront créés. Dans
                l'éventuaité où aucun point n'est créé et
                que la liste ne dépasse pas la taille
                demandée, on rajoute simplement un
                compteur assurant la terminaison de l'
                algorithme, d'où l'utilité de ce "m" créé
                en début d'algorithme.

                 nvabscisses.append((abscisses[k]+
                     abscisses[k+1])/2)
                 nvordonnees.append((ordonnees[k]+
                     ordonnees[k+1])/2)

             nvabscisses.append(abscisses[k+1])
             nvordonnees.append(ordonnees[k+1])

         abscisses=nvabscisses
         ordonnees=nvordonnees
         m-=1

    return (abscisses,ordonnees)




"""
En entrée : la coordonnée considérée.
En sortie : l'une des deux coordonnées du carreau auqul
    elle appartient

C'est une simple dichotomie, à la différence près qu'lle
    ne retroune pas un encadrement de notre coordonnée de
    point mais bel et bien le carreau auquel elle
    appartient.
"""
def dicho(x):

    kmin=1
    kmax=100000

    while abs(kmax-kmin)>1:
```

```
            xkmin=kmin*0.1
            xkmax=kmax*0.1
            i=(kmax+kmin)//2
            xi=i*0.1

            if xi>=x:
                kmax=i

            else:
                kmin=i

        return kmax




"""
Ceci est la première version du "nombre de carreaux touchés" par la courbe, qui permet de visualiser le résultat à la fin mais qui limite énormément le facteur de zoom du flocon faute de mémoire de l'ordinateur.
Elle se base sur un principe relativement simple : on crée une matrice nulle "touche " de taille 1000 par 1000 (on en crée en fait 2, mais la matrice "mat" ne sert qu'à la visualisation du résultat) chaque case [i,j] codant pour le carré compris dans la zone du plan ([0.01*(i-1),0.01*i[ et [0.01*(j-1),0.01*j[, dont on met à jour le coefficient (1) lorsqu'on sait qu'un point est passé par ce carreau.
Cette version de la fonction retourne non seulement le nombre de carreaux touchés mais aussi une représentation graphique du nombre de carreaux touchés.
Exeptionnellement, il est préférable pour cet algorithme de créer le flocon entre les points (0,0) et (50,0) pour obtenir un résultat visuel correct.
"""
def remplissage1(ab,ordo,n):

    table=CreationPtsIntermediaires(ab,ordo,n)
    absc=table[0]
    ordo=table[1]
    touche=np.zeros((1000,1000))
    mat=np.zeros((1000,1000,3))

    for k in range(len(absc)):
        i=dicho(absc[k])
        j=dicho(ordo[k])
        touche[i,j]=1
        mat[j,i,0]=1
```

```
    plt.imshow(mat)
    plt.show()

    return np.sum(touche)



"""
Même principe que la fonction remplissage1, à la diffé
    rence que dans ce cas-ci, on peut créér une matrice
    peaucoup plus grande vu qu'on n'a pas besoin de coder
    la représentation graphique.
"""
def remplissage2(ab,ordo,n):

    table=CreationPtsIntermediaires(ab,ordo,n)
    absc=table[0]
    ordo=table[1]
    touche=np.zeros((10000,10000))

    for k in range(len(absc)):
        i=dicho(absc[k])
        j=dicho(ordo[k])
        touche[i,j]=1

    return np.sum(touche)



"""
En entrée : une liste (dans notre cas, de coordonnées) et
     notre coefficient de dilatation des valeurs (pour é
    muler l'agrandissement du flocon).
En sortie : la même liste avec des coefficients dilatés
    du facteur souhaité.
"""
def dilatation(a,facteur):

    newa=[]
    for i in range(len(a)):
       newa.append(facteur*a[i])
    return newa
```

```
"""
Cette fonction est une fonction "synthèse".
En entrée : les listes des abscisses et ordonnées des 2
    points initiaux de notre flocon et notre niveau de pré
    cision n.
En sortie : la liste des facteurs de dilatation du flocon
     et la liste des carreaux touchés pour chaque facteur
    de dilatation

NB : On choisit arbitrairement les coefficients de
    dilataion
"""
def ListeResultatsRemplissage(ab,ordo,n):

    listeFacteurs=[k for k in range(1,100,3)]
    nbCarreaux=[remplissage2(ab,ordo,n)]

    for k in listeFacteurs[1::]:
        a=dilatation(ab,k)
        b=dilatation(ordo,k)
        nbCarreaux.append(remplissage2(a,b,n))

    return listeFacteurs,nbCarreaux



"""
Situation : nous avons maintenant à notre disposition la
    liste des facteurs de dilatation du flocon et la liste
     des carreaux touchés pour chaque facteur de
    dilatation, qui prend l'allure d'une courbe d'équation
     y=c*(x**s) (avec c un coefficient qui ne nous importe
     pas): c'est cet exposant "s" qui va nous fournir la
    dimension recherchée. POur faciliter sa recherche, on
    passe au log pour obtenir une droite d'équation y= log
    (c)+s*log(x). Il suffit alors d'évaluer le coefficient
     directeur de cette courbe pour trouver s.

Le but de cette fonction est donc simplement de tracer
    ladite droite avec la liste des facteurs et des
    carreaux dont elle dispose.
"""
def CalculDeDimension(ab,ordo,n):

    ListeFacteursZoom=ListeResultatsRemplissage(ab,ordo,n
        )[0]
    ListeCarreaux=ListeResultatsRemplissage(ab,ordo,n)[1]

    LogFacteurs=[]
```

```
    LogCarreaux=[]

    for i in range(len(ListeFacteursZoom)):

        LogFacteurs.append(np.log(ListeFacteursZoom[i]))
        LogCarreaux.append(np.log(ListeCarreaux[i]))

    plt.plot(LogFacteurs,LogCarreaux)
    plt.show()

    return LogFacteurs,LogCarreaux

"""
CONCLUSION :

ListeResultatsRemplissage retourne [1, 4, 7, 10, 13, 16,
    19, 22, 25, 28, 31, 34, 37, 40, 43, 46, 49, 52, 55,
    58, 61, 64, 67, 70, 73, 76, 79, 82, 85, 88, 91, 94,
    97], [13.0, 97.0, 213.0, 325.0, 471.0, 625.0, 785.0,
    953.0, 1127.0, 1423.0, 1545.0, 1753.0, 1971.0, 2197.0,
     2349.0, 2553.0, 2749.0, 2917.0, 3207.0, 3475.0,
    3703.0, 3929.0, 4165.0, 4477.0, 4551.0, 4903.0,
    5115.0, 5531.0, 5749.0, 5950.0, 6031.0, 6343.0,
    6658.0]

CalculDeDimension retourne [0.0, 1.3862943611198906,
    1.9459101490553132, 2.302585092994046,
    2.5649493574615367, 2.772588722239781,
    2.9444389791664403, 3.091042453358316,
    3.2188758248682006, 3.332204510175204,
    3.4339872044851463, 3.5263605246161616,
    3.6109179126442243, 3.6888794541139363,
    3.7612001156935624, 3.828641396489095,
    3.8918202981106265, 3.9512437185814275,
    4.007333185232471, 4.060443010546419,
    4.110873864173311, 4.1588830833596715,
    4.204692619390966, 4.248495242049359,
    4.290459441148391, 4.330733340286331,
    4.3694478524670215, 4.406719247264253,
    4.442651256490317, 4.477336814478207,
    4.51085950651685, 4.543294782270004,
    4.574710978503383], [2.5649493574615367,
    4.574710978503383, 5.3612921657094255,
    5.783825182329737, 6.154858094016418,
    6.437751649736401, 6.665683717782408,
    6.859614903654202, 7.027314514039777,
    7.260522598089852, 7.3427791893318455,
    7.469083884921234, 7.58629630715272, 7.69484807238461,
     7.761744984658913, 7.845024417241484,
    7.918992488165245, 7.978310969867722,
```

```
    8.073091199693154, 8.153349757998892,
    8.216898580913613, 8.27614021955846,
    8.334471554600944, 8.406708458240965,
    8.423102268016642, 8.497602541651233,
    8.539932678385727, 8.618123909994678,
    8.656781205623291, 8.691146498539675,
    8.704668113450987, 8.755107121633896,
    8.80357441813497]
"""


#####################################
#####################################
###########                 ###########
###########    s=1,29       ###########
###########                 ###########
#####################################
#####################################

"""
La valeur exacte à 10e-3 près est de 1,262 : on obtient
    un écart relatif de 2%, et cela laisse penser que la
    valeur converge effectivement vers 1.262 à force d'ité
    rer les dilatations.
"""
```

6.3 Annexe 3 : programme Python et pour afficher l'ensemble de Mandelbrot

Ci-après le programme permettant d'afficher **l'ensemble de Mandelbrot**

```
import math as mt
import numpy as np
import matplotlib.pyplot as plt



def couleur(n):

    #On fait ici le choix de ne pas prendre plusieurs
        paramètres égaux pour la couleur, sans quoi nous
        obtiendrions différentes teintes de gris.

    return [n/200, 1-(n/200)]
```

```
def mandelbrot(abs1,abs2,ord1,ord2,N,T,e):
    #abs1, abs2, ord1 et ord2 permettent de choisir le
        cadre d'étude de l'ensemble. N désigne nombre de
        pixels de la verticale. T désigne le nombre d'ité
        ration que l'on effectue sur la suite pour vé
        rifier que son module ne dépasse pas 2. Enfin, e d
        ésigne l'exposant du terme en u_n, dans le calcul
        de u_n+1. Dans l'ensemble de Mandelbrot classique,
         on a e=2.
    #c permet de recalibrer le tableau en fonction des
        paramètres demandés, dans le cas où la zone ciblée
         n'est pas un carré

    c=int(abs(abs2-abs1)*N/abs(ord2-ord1))
    tab=np.zeros((N,c,3), dtype=float)

    #f et g désignent le pas vertical et le pas
        horizontal qui permettront de parcourir la zone
        ciblée, ils seront d'autant plus faible que le
        nombre de pixels souhaités sera important.

    f=(ord1-ord2)/N
    g=(abs2-abs1)/c

    #On parcourt à présent le tableau

    for b in range(c):
        for a in range(N):

            #On réactualise la pseudo-raison de la
                nouvelle suite

            z=-1j*(ord2+a*f)+(abs1+b*g)
            d=z
            p=0

            #On fixe comme test d'arrêt le dépassement de
                 la valeur 2 pour le module de la suite,
                pour un nombre d'itération inférieur à T.

            while p<T and abs(d)<2:
                p+=1
                d=(d**e)+z

            #Si p=T, alors la suite a pu atteindre sa T-è
                me valeur sans dépasser la barrière de
                module 2. Si T est assez grand, on peut
                supposer que la suite ne divergera pas. On
                 colorie ainsi le pixel en noir.
```

```
            if p==T:
                tab[a,b,0],tab[a,b,1],tab[a,b,2]=0,0,0

            #Si en revanche, p<T, alors dans ce cas, la
                suite a vu son module dépasser 2 à un
                moment. On en déduit qu'elle n'est pas
                bornée. On colorie le pixel en fonction de
                 l'écart entre T et p, c'est à dire en
                fonction de la vitesse à laquelle son
                module a atteint la valeur 2.

            else:

                tab[a,b,0],tab[a,b,1],tab[a,b,2]=couleur(
                    T-p)[1],0,couleur(T-p)[0]

    plt.imshow(tab)
    plt.show()




def julia(abs1,abs2,ord1,ord2,N,T,z_0):

    #Les conventions utilisées sont les mêmes que pour l'
        algorithme précedent

    c=int(abs(abs2-abs1)*N/abs(ord2-ord1))
    tab=np.zeros((N,c,3), dtype=float)
    f=(ord1-ord2)/N
    g=(abs2-abs1)/c

    for b in range(c):
        for a in range(N):

            z=1j*(ord2+a*f)+(abs1+b*g)
            d=z
            p=0

            while p<T and abs(d)<2:
                p+=1

                #Ici, la pseudo-raison de la suite n'est
                    pas z, mais z_0, argument de la
                    fonction, contrairement à Mandelbrot.
                    Ici, c'est le terme initial de la
                    suite qui permet de caractériser un
                    point dans la représentation, à z_0
```

```
                    fixé.

                d=(d**2)+z_0

            if p==T:
                tab[a,b,0],tab[a,b,1],tab[a,b,2]=0,0,0

            else:
                tab[a,b,0],tab[a,b,1],tab[a,b,2]=couleur(
                    T-p)[1],couleur(T-p)[0],0

    plt.imshow(tab)
    plt.show()
```

6.4 Annexe 4 : programme Python pour afficher la courbe remplissant l'espace

Ci-après le programme permettant d'afficher la **courbe de Peano**.

```
import math
import numpy as np
import matplotlib.pyplot as plt
import matplotlib

'''
Retourne la valeur de f en t
'''
def f_function(t):
    #f est 1-périodique, donc on récupère la partie
        fractionnaire de t
    t = math.modf(t)[0]

    if t <= 0.4:
        return 0

    if t > 0.4 and t <= 0.5:
        return 10*t - 4

    if t > 0.5 and t <= 0.8:
        return 1

    if t > 0.8 and t <= 0.9:
        return -10*t + 9

    return 0

'''
```

```
Retourne la valeur de g en t
'''
def g_function(t):
    #Même commentaire que pour f
    t = math.modf(t)[0]

    if t <= 0.2:
        return 0

    if t > 0.2 and t <= 0.3:
        return 10*t - 2

    if t > 0.3 and t <= 0.4:
        return 1

    if t > 0.4 and t <= 0.5:
        return -10*t + 5

    if t > 0.5 and t <= 0.6:
        return 0

    if t > 0.6 and t <= 0.7:
        return 10*t - 6

    if t > 0.7 and t <= 0.8:
        return 1

    if t > 0.8 and t <= 0.9:
        return -10*t + 9

    return 0

'''
Retourne l'abscisse du point à l'instant t. N définit
    jusqu'où la somme est indexée
'''
def x(t, N):
    res = 0

    for n in range(1, N + 1):
        numerateur = f_function( (10**(n-1)) * t)
        denominateur = 2**n
        res += (numerateur / denominateur)

    return res

'''
Retourne l'ordonnée du point à l'instant t. N définit
    jusqu'où la somme est indexée
'''
```

```
def y(t, N):
    res = 0

    for n in range(1, N + 1):
        numerateur = g_function( (10**(n-1)) * t)
        denominateur = 2**n
        res += (numerateur / denominateur)

    return res

'''
Retourne l'ordre d'erreur permettant qu'elle soit
    invisible sur l'écran (en fonction de la largeur en
    pixels de celui-ci)
'''
def majorer_erreur(nb_pixels):
    n = 0
    majore = False

    while not majore:
        n += 1
        if 2**n >= nb_pixels:
            majore = True

    return n

'''
Affiche la courbe, avec le pas de temps spécifié et le
    nombre de pixels qu'a l'écran en largeur
'''
def afficher(pas, nb_pixels):
    #On récupère l'indexation maximale des sommes et on
        initialise toutes les variables
    N = majorer_erreur(nb_pixels)
    abscisses = []
    ordonnees = []

    t = 0

    while t <= 1:
        #A chaque étape, on récupère l'abscisse et l'
            ordonnée, et on incrémente le temps
        x_local = x(t, N)
        y_local = y(t, N)

        abscisses.append(x_local)
        ordonnees.append(y_local)

        t += pas
```

```
    #On affiche le résultat
    plt.plot(abscisses, ordonnees)
    plt.axis("equal")

    plt.show()

'''
Anime la construction de la courbe, avec le pas de temps
   spécifié et le nombre de pixels qu'a l'écran en
   largeur
'''
def animer(pas, nb_pixels):
    #On récupère l'indexation maximale des sommes et on
        initialise toutes les variables
    N = majorer_erreur(nb_pixels)
    abscisses = [x(0, N)]
    ordonnees = [y(0, N)]

    t = pas
    graphe, = plt.plot(abscisses, ordonnees)
    plt.axis([0, 1, 0, 1])

    while t <= 1:
        #A chaque étape, on récupère l'abscisse et l'
            ordonnée, et on incrémente le temps
        x_local = x(t, N)
        y_local = y(t, N)

        abscisses.append(x_local)
        ordonnees.append(y_local)

        graphe.set_xdata(abscisses)
        graphe.set_ydata(ordonnees)

        plt.pause(0.01)

        t += pas

    #On affiche le résultat
    plt.axis([0, 1, 0, 1])
    plt.show()
```

Chapitre 7

Bibliographie

3Blue1Brown, Fractals are typically not self-similar : disponible sur https://www.youtube.com/watch?v=gB9n2gHsHN4

Topologie Générale : Connexité : disponible sur https://fr.wikiversity.org/wiki/Topologie_g%C3%A9n%C3%A9rale/Connexit%C3%A9

Bertrand Rémy, Continuité et Compacité : disponible sur http://bremy.perso.math.cnrs.fr/MAT311-2017-SlidesAmphi2-ContinuiteEtCompacite-Re%CC%81sume%CC%81.pdf

Jean-Louis Rouget, Topologie des espaces vectoriels normés : disponible sur http://www.maths-france.fr/MathSpe/Cours/13-topologie.pdf

Courbe de Peano : disponible sur https://fr.wikipedia.org/wiki/Courbe_de_Peano

Progression de la courbe de Peano : disponible sur https://skullsinthestars.files.wordpress.com/2014/03/peanoprogression.jpg

Triangle de Sierpinski : disponible sur https://fr.wikipedia.org/wiki/Triangle_de_Sierpi%C5%84ski

Créer des fractales en Python à l'aide du module Turtle : disponible sur https://www.lespritsorcier.org/blogs-membres/module-turtle-de-python/

Xavier Gourdon, Les Maths en Tête (Analyse) : éditions *Ellipses*

O. Knill, The Mandelbrot Set : disponible sur http://abel.math.harvard.edu/archive/118r_spring_05/handouts/mandelbrot.pdf

L'ensemble de Mandelbrot : disponible sur https://fr.wikipedia.org/wiki/Ensemble_de_Mandelbrot

Espaces topologiques : disponible sur http://math.univ-lyon1.fr/~brandolese/enseignement/L3topologie/poly2008.pdf

Antoine Diez, Connexité. Exemples et applications : disponible sur `http://perso.eleves.ens-rennes.fr/people/Antoine.Diez/connexite.pdf`

Thibaut Deheuvels, Mesures et dimension de Hausdorff, Introduction à l'étude des ensembles auto-similaires : disponible sur `https://w3.ens-rennes.fr/math/people/thibaut.deheuvels/Mesures-Hausdorff.pdf`