BORNE Florentin
FORTIN Raphaël
GAUDIN Mateo
VISOMBLAIN Eva

IPSA
ÉCOLE D'INGÉNIEURS
DE L'AIR, DE L'ESPACE ET DE LA MOBILITÉ DURABLE

OPTIMISATION DIFFÉRENTIABLE 2

BE Ma324

Méthode de Newton et SVM

Jeudi 11 mai 2023

M. COUFFIGNAL – M. EL-MAHBOUBY

Dans ce Bureau d'Études, nous étudierons les fractales avec la Méthode de Newton, une des méthodes algorithmiques de résolution d'équations, ainsi que la méthode SVM (Support-Vector Machine) avec noyau, une méthode alternative aux réseaux de neurones.

Les fractales sont des objets mathématiques complexes qui ont une structure répétitive à différentes échelles. Elles ont été introduites pour la première fois par le mathématicien français Benoit Mandelbrot dans les années 1970.

Les fractales sont souvent créées en utilisant des algorithmes récursifs, c'est-à-dire des règles qui se répètent à différentes échelles. Par exemple, le célèbre ensemble de Mandelbrot est créé en appliquant une formule mathématique simple à des nombres complexes à plusieurs reprises.

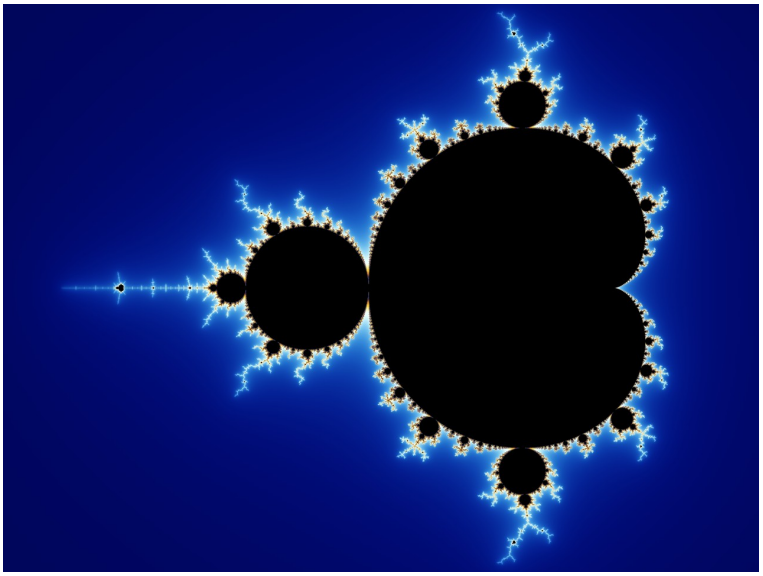

FIGURE 1 – Ensemble de Mandelbrot, exemple de fractale

Table des matières

Première partie

Des fractales avec la méthode de Newton

Dans cette partie, nous étudierons la méthode de Newton, tout d'abord dans le plan complexe, puis à l'aide de représentations graphiques.

La méthode de Newton est une méthode de résolution de l'équation $f(x) = 0$. Si x_0 est proche de la racine $\overline{x}$ on peut faire un développement de Taylor à l'ordre 1 de la fonction f en x_0 :

$$f(x) = f(x_0) + (x - x_0)f'(x_0) + O\big((x - x_0)^2\big)$$

Pour trouver une valeur approchée de $\overline{x}$, on ne garde que la partie linéaire du développement :

$$f(\overline{x}) = 0 \simeq f(x_0) + (\overline{x} - x_0)f'(x_0)$$

puis si $f'(x_0) \neq 0$ on obtient finalement :

$$\overline{x} = x_0 - \frac{f(x_0)}{f'(x_0)}$$

Graphiquement, cela revient à tracer la tangente à la courbe représentative de f et à chercher où elle coupe l'axe des abscisse. On considère donc la suite récurrente définie par une valeur x_0 proche de la racine et par la relation :

$$x_{n+1} := x_n - \frac{f(x_n)}{f'(x_{n+1})}$$

1 Méthode de Newton dans le plan complexe

Étant donné un nombre complexe a fixé, on considère la fonction polynomiale :

$$P(x) = (x - 1)(x + \frac{1}{2} - a)(x + \frac{1}{2} + a)$$

L'algorithme de Newton associé à P est le suivant :

x_0 donné
$x_1 = x_0 - \dfrac{P(x_0)}{P'(x_0)}$
While $x_{k+1} - x_k > \epsilon$
$x_{k+1} = x_k - \dfrac{P(x_k)}{P'(x_k)}$

2 Représentation graphique

La suite $(x_n)_{n\in\mathbb{N}}$ engendrée par l'algorithme de Newton va en général converger vers l'une des trois racines de P. Cependant, dans certains cas, elle ne va pas converger.

Nous allons représenter une portion carrée du plan complexe :

$$(\mathscr{R}e(z), \mathscr{I}m(z)) \in [u, v]^2$$

dans laquelle chaque point d'affixe z sera coloré d'une façon différente suivant que la suite $(x_n)_{n\in\mathbb{N}}$ converge vers 1, $-\frac{1}{2} - a$, $-\frac{1}{2} + a$ ou ne converge pas.

Nous nous sommes donc affairés à rédiger un code qui nous permettait d'afficher cette matrice $n \times n$. Par ailleurs, chaque élément de cette matrice correspond à une valeur $x_0 \in C$ qui, après un passage dans l'algorithme de Newton, prendra la valeur 0, 1, 2 ou 3. Ces chiffres correspondent aux différentes convergences possibles, et cela nous a permis d'afficher une image.

Nous avions fait plusieurs versions de ce code car il n'était pas très optimisé à la base. Enfin, nous sommes arrivés à une version nous permettant d'afficher des matrices 500×500 en trois secondes, la rapidité étant l'un des objectifs que nous nous étions fixés.

Voici l'affichage que nous avons avec la fractale de Newton, obtenue lorsque $a := i\frac{\sqrt{3}}{2}$, c'est-à-dire lorsque $P(x) = x^3 - 1$. Nous l'avons représenté pour l'ensemble donné dans le sujet :

$$\{z \in \mathbb{C}/|(\mathscr{R}e(z)| \leq 2, |\mathscr{I}m(z)| \leq 2\}$$

avec une taille de 10000×10000.

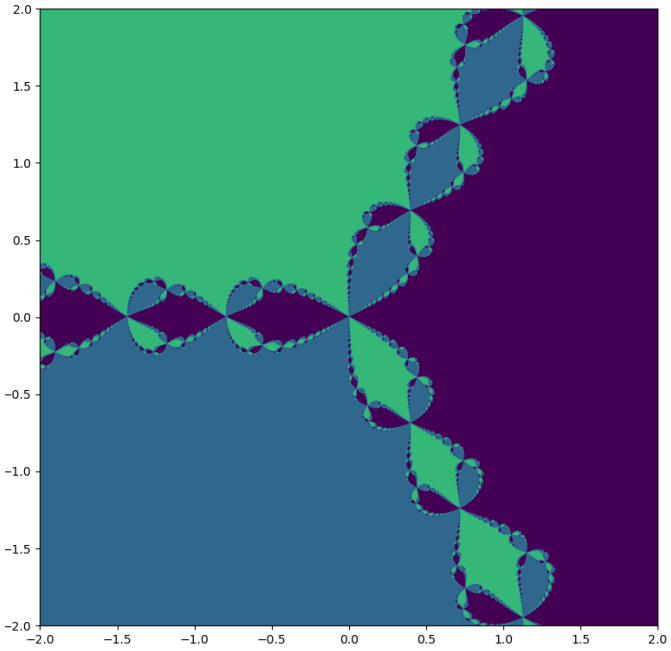

FIGURE 2 – Fractale de Newton 10000×10000

Par ailleurs, le sujet nous livre la valeur $a = -0,00508 + 0,33136i$, qui nous permet d'observer un phénomène intéressant, la présence d'un lapin de Douady. Nous nous sommes donc tentés à l'afficher dans les domaines :

$$\{z \in \mathbb{C}/|(\mathscr{R}e(z)| \leq 2, |\mathscr{I}m(z)| \leq 2\} \text{ et } \{z \in C/|(\mathscr{R}e(z)| \leq \frac{1}{10}, |\mathscr{I}m(z)| \leq \frac{1}{10}\}$$

Voici les résultats obtenus :

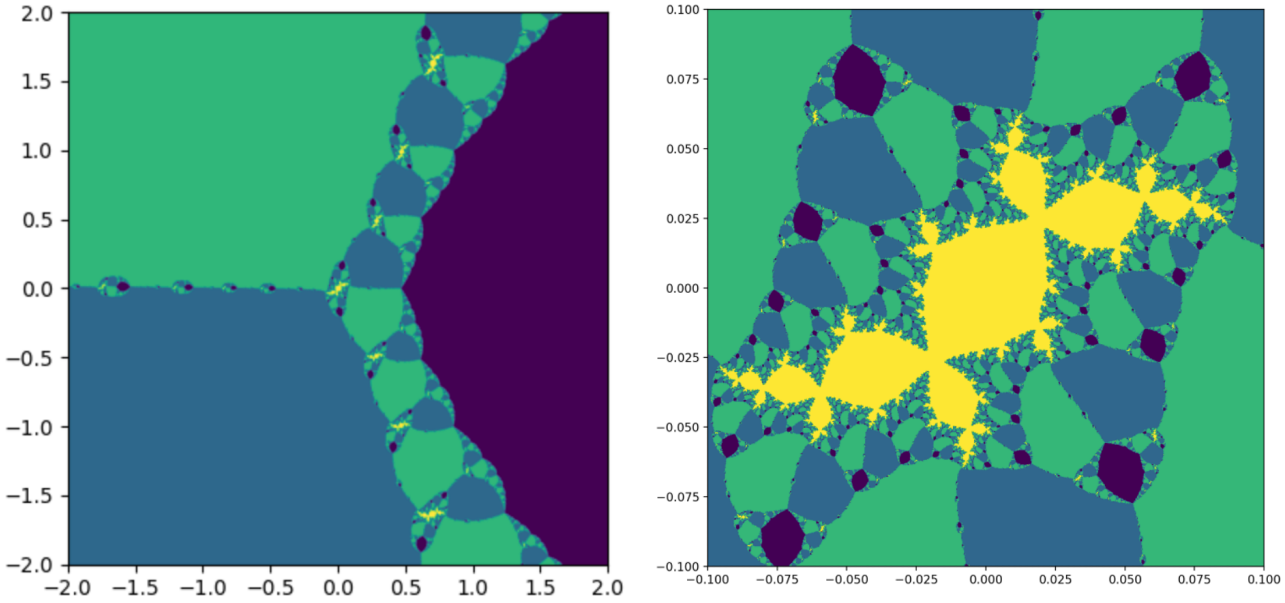

FIGURE 3 – Fractale Douady 1 et 2 10000 × 10000

Ensuite, il nous a été demandé de réaliser une vidéo qui zoome sur les différentes fractales. Nous avons donc créé une première fonction qui zoomait petit à petit sur une fractale donnée. Cette première fonction admettait plusieurs défauts. En effet, nous ne pouvions zoomer qu'au centre de l'image. Nous avons développé une deuxième version de ce code qui nous permettait de zoomer sur un point défini de la fractale.

Cette deuxième version admettait encore un défaut : Où zoomer exactement pour obtenir un résultat vidéo ? Car en effectuant plusieurs tests sur des points approximatifs, on se retrouvait au bout de quelques itérations dans une seule couleur.

Nous avons donc développé une fonction "LeKlik", qui porte certes une dénomination douteuse, mais qui nous permet de zoomer manuellement et progressivement directement sur l'interface **Matplotlib**. En somme, la fractale de notre choix s'affiche, et il suffit à l'utilisateur de cliquer sur la fractale pour obtenir un zoom à cet endroit. On peut ainsi zoomer autant de fois qu'on le souhaite sur la fractale, jusqu'à atteindre le point le plus lointain possible. Une fois ce point pris en note, il suffit de le mettre comme point central dans la fonction vidéo, qui nous créera un zoom progressif jusqu'à ce point. Alors nous passons ici les détails, mais il faut bien sûr renseigner le facteur de zoom désiré, la résolution de l'image, l'intervalle de valeur, le nombre d'images par seconde de la vidéo, etc.

Après de nombreux essais avec cette fonction, on a noté qu'elle n'était pas précise tout le temps. Nous n'avons pas réussi à la modifier de sorte à ce qu'elle le soit tout le temps. En l'occurrence, les clics sont très efficaces aux quatre coins ainsi qu'au centre de l'image. Cela nous a tout de même permis de trouver des points très précis, et après avoir compris le fonctionnement, il est aisé de naviguer dans les fractales.

Nous sommes, de cette manière, parvenus à zoomer à des points avec 11 décimales (maximum de précision obtenu) avec comme point central : $x = -0,05583079871$ et $y = 0,50654650778$ pour le polynôme obtenir lorsque $a = -0,001+0,7i$. Au-delà de cette précision, nous n'avions quasiment plus rien à observer que des sortes de tâches de couleurs. Après réflexion, nous avons finalement compris que ces taches de couleurs étaient en fait dues au nombre d'itérations que nous faisions dans la fonction fractale, le manque de précision se répercute sur l'image à des échelles minimes.

Plusieurs de ces vidéos sont présentes dans notre fichier de rendu et sont accessibles sur Youtube. A ce titre, nous avons fait des tests avec de nombreux autres polynômes, en faisant varier tous les paramètres comme l'intervalle, les polynômes, les couleurs, ainsi que les effets vidéos (zoom, FPS, qualité, etc.).

Voici les 3 liens de nos vidéos YouTube : Vidéo 1 ; Vidéo 2 ; Vidéo 3

A ce titre, vous trouvez des tests effectués avec de nombreux autres polynômes, en voici quelques exemples :

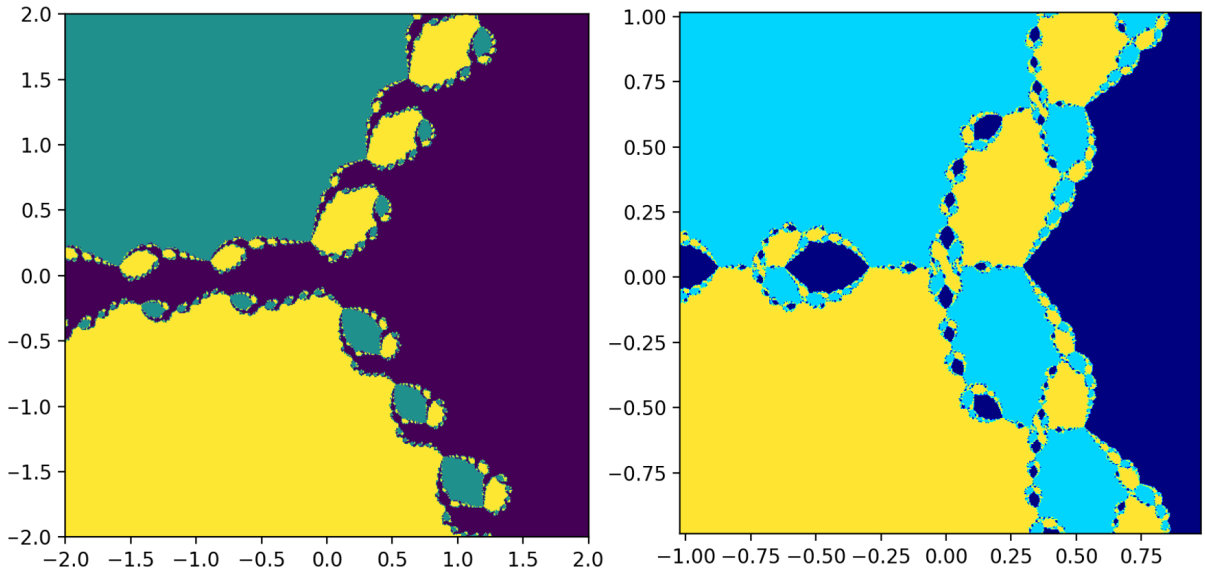

FIGURE 4 – Fractales obtenues avec $a = -0,1011 + 0,9563i$ et $a = 0,001 + 0,7i$

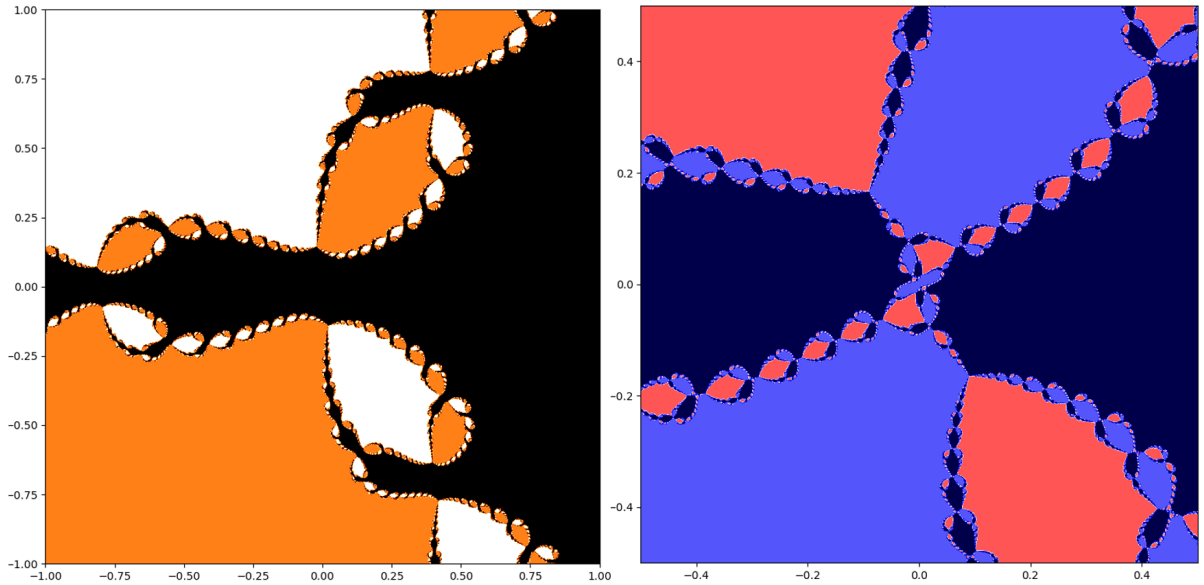

FIGURE 5 – Fractales obtenues avec $a = 0,01 + 0,9i$ et $a = 0,05 + 0,9i$