

# Travail pratique de programmation séquentielle

## Les ensembles de Julia

### 1 Objectif

- Programmer en s'amusant.
- Utilisation de tableau bidimensionnels.
- Utilisation de structures.
- Utilisation de la librairie SDL.
- Utilisation des nombres complexes.
- Afficher un fractal: l'ensemble de Julia.

### 2 Ensembles de Julia

Les ensembles de Julia (du mathématicien français Gaston Julia, 1893-1978) sont des objets fractals du plan complexe (plus sur les complexes à la fin du document). Dans ce contexte, le mot fractal signifie que ces objets ont une dimension fractionnaire et présentent des invariances d'échelle. Pour définir les ensembles de Julia, on considère la famille de fonctions complexes

$$P_c : \mathbb{C} \rightarrow \mathbb{C}, \quad (1)$$

$$z \rightarrow z^2 + c, \quad (2)$$

où  $c \in \mathbb{C}$ .

On dénote  $K_c$ , l'ensemble des points  $z \in \mathbb{C}$ , tels que la suites des itérés  $P_c^n(z)$  est bornée:

$$K_c = \{z \in \mathbb{C} \mid \sup |P_c^n(z)| < \infty\}, \quad (3)$$

où  $P_c^n(z)$  est défini par

$$P_c^n(z) = P_c(P_c(\dots P_c(z))). \quad (4)$$

L'ensemble de Julia  $J_c$  est le bord de  $K_c$ . En d'autres termes, s'il existe  $M \in \mathbb{R}$  tel que  $|P_c^n(z)| \leq M$ . Dans ce TP, nous considérons  $M = 2$ .

Les ensembles de Julia ont quelques propriétés importantes

1.  $J_c$  est inclu dans le cercle de centre 0 et de rayon  $|c| + 1$ .
2.  $J_c$  est symétrique par rapport à 0.
3.  $J_{\bar{c}}$  est le symétrique de  $J_c$  par rapport à l'axe des réels. En particulier, si  $c$  est réel,  $J_c$  est symétrique par rapport à l'axe des réels.

Comme  $J_c$  est inclus dans le disque de rayon  $|c| + 1$  centré en 0, on peut affirmer que  $z \notin J_c$  si et seulement si la suite des itérés  $P_c^n(z)$  sort de ce disque à une itération donnée.

### 3 Calcul d'un ensemble de Julia

Pour calculer l'ensemble de Julia,  $J_c$ , associé à une valeur,  $c \in \mathbb{C}$ , on peut se restreindre au domaine carré  $[-2, 2] \times [-2i, 2i] \in \mathbb{C}$ . Ce domaine est discrétisé en une grille dont la finesse du maillage déterminera la précision du calcul. La grille peut être stockée sous forme d'un tableau d'entiers. L'élément du tableau correspondant à un  $z$  donné, contient l'itération,  $k$ , à partir de laquelle  $P_c^k(z)$  sort du disque de rayon 2. C'est cette valeur,  $k$ , que nous souhaitons calculer et afficher dans ce travail.

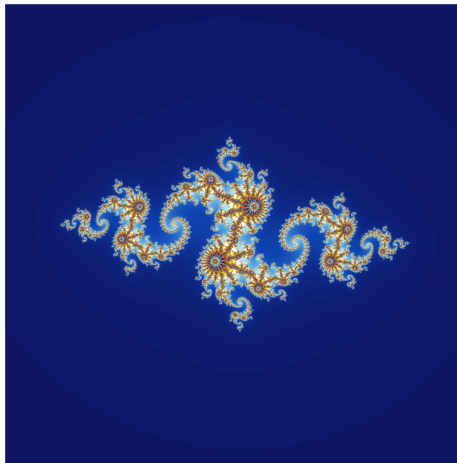


FIGURE 1 – Exemple d'un ensemble de Julia pour  $c = -0.8 + 0.156i$ . Source wikipedia: <https://bit.ly/3i1g4QC>

### 4 Travail à réaliser

Le calcul de l'ensemble  $K_c$  associé à  $c \in \mathbb{C}$ , se fait sur le carré du plan complexe, dont le coin inférieur gauche est  $-2 - 2i$  et supérieur droit  $+2 + 2i$ , carré qu'on discrétise en une grille. Pour chacune des valeurs complexes de la grille, appelons les  $z_0$ , on détermine si la suite des itérés

$$z_{n+1} = P_c(z_n) = z_n^2 + c, \quad (5)$$

est bornée ou non. Une bonne heuristique est obtenue en effectuant un certain nombre d'itérations, `nb_iter`, pour voir si la suite sort du disque de rayon 2. Ceci correspond à la condition

$$|z_n| \leq 2. \quad (6)$$

Évidemment, `nb_iter` représente une borne maximale, car on peut interrompre l'itération dès qu'on a quitté le disque de rayon 2. En d'autres termes, on enregistre le  $n$ , tel que  $z_n > 2$ . Si  $n > \text{nb\_iter}$  on interrompt la boucle et on enregistre ou  $n = \text{nb\_iter}$ .

Visualisez les résultats à l'aide de la librairie `SDL2`. Quelques fonctions ainsi qu'un exemple d'utilisation se trouvent sur `cyberlearn` (dans l'archive

`gfx_example.tar`<sup>1</sup>) ou directement en cliquant sur [ce lien](#). Je **répète ici**, affichez l'itération,  $k$ , à laquelle  $z_k > 2$  en niveaux de gris (dans ce cas là, chaque canal de couleur a la même intensité, mais si vous avez du temps à perdre vous pouvez construire vos propres colormaps). Vous pouvez aller sur la page <https://bit.ly/2HdQ6Jb> pour trouver des exemples de valeurs de  $c$  et les résultats attendus.

Affichez également une jolie animation en faisant varier  $c$  en prenant

$$c = 0.7885 \cdot (\cos(\theta) + i \cdot \sin(\theta)), \quad (7)$$

où  $\theta \in [0, 2\pi[$ . Il s'agit donc ici de faire la boucle d'affichage suivante:

1. Calculer et afficher l'ensemble de Julia pour  $\theta_0 = 0$ .
2. Calculer  $\theta_1 = \theta_0 + \delta\theta$ , avec  $\delta\theta$  un petit paramètre.
3. Recalculer l'ensemble de Julia et l'afficher avec  $\theta_1$ .
4. Recommencer l'opération avec  $\theta_2 = \theta_1 + \delta\theta$ ,  $\theta_3 = \theta_2 + \delta\theta, \dots, \theta_i = \theta_{i-1} + \delta\theta$ .

## 4.1 Nombres complexes

Comme vous l'aurez compris pour réaliser ce travail il faut réaliser une librairie de manipulation de nombres complexes. Chaque nombre complexe est constitué une paire de nombre réels: la partie **réelle** et la partie **imaginaire** du nombre. Le type complexe sera donc une structure

```
typedef struct _complex {
    double re, im; // notés a et b plus bas
} complex;
```

Ensuite il faut implémenter les règles d'addition, de multiplication entre deux nombres complexes, ainsi que le module d'un nombre complexe.

Soit  $z \in \mathbb{C}$  un nombre complexe. On note ce nombre

$$z = a + i \cdot b, \quad (8)$$

où  $a$  est la partie **réelle** de  $z$ ,  $b$  la partie **imaginaire** de  $z$ , et  $i$  le nombre imaginaire qui a la super propriété  $i^2 = -1$ . Avec cette notation, on peut calculer presque comme avec des nombres réels. Ainsi l'addition de deux nombres complexes,  $z_1 = a_1 + i \cdot b_1$ ,  $z_2 = a_2 + i \cdot b_2$  s'écrit

$$z_1 + z_2 = (a_1 + a_2) + i \cdot (b_1 + b_2). \quad (9)$$

Le produit est un peu plus complexe mais s'écrit

$$z_1 \cdot z_2 = (a_1 \cdot a_2 - b_1 \cdot b_2) + i \cdot (a_1 b_2 + a_2 b_1). \quad (10)$$

Finalement le module de  $z = a + i \cdot b$  s'écrit

$$|z| = \sqrt{a^2 + b^2}. \quad (11)$$

---

1. La librairie SDL ne libère pas bien la mémoire. Les sanitizers vont donc afficher des erreurs lors de la fin de votre programme. C'est normal.

Comme vous pouvez le voir dans la définition de la structure des nombres complexes, à aucun moment nous n'avons besoin de représenter  $i$  dans notre structure de données.  $i$  est nécessaire pour les humains pour qu'ils puissent calculer plus facilement avec les nombres complexes.

Avec ce qui précède vous pouvez implémenter les fonctions

```
complex complex_add(complex lhs, complex rhs);  
complex complex_mul(complex lhs, complex rhs);  
double complex_module(complex lhs);
```

et avez en votre possession tout ce qu'il faut pour écrire le code qui vous permettra de faire des figures plus incroyables les unes que les autres.