Travail pratique de programmation concurrente

Travail pratique - Les ensembles de Julia

1 Objectif

Se familiariser avec les threads POSIX sur un problème simple et mesurer le gain en performance en utilisant plusieurs threads.

2 Ensembles de Julia

Les ensembles de Julia (du mathématicien français Gaston Julia, 1893-1978) sont des objets fractals du plan complexe. 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

$$P_c : \mathbb{C} \to \mathbb{C}, \tag{1}$$

$$z \to z^2 + c, \tag{2}$$

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:

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

où $P_c^n(z)$ est défini par

$$P_c^n(z) = P_c(P_c(...P_c(z))). \tag{4}$$

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 $P_c^n(z)$ sort du disque de rayon 2. C'est cette valeur, k, que nous souhaitons calculer et afficher dans ce travail.

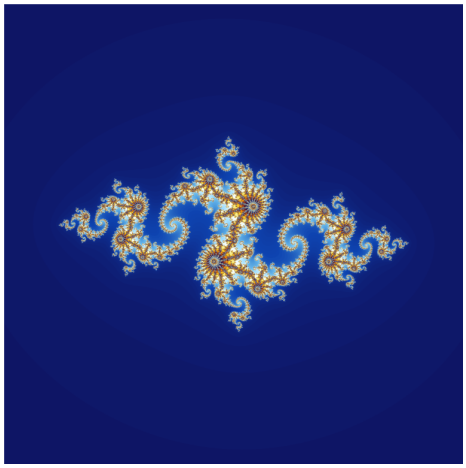

Figure 1 – Exemple d'un ensemble de Julia pour $c = -0.8 + 0.156i$. Source wikipedia: https://bit.ly/2tY6mp2

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

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

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

$$|z_n| \leq 2. \tag{6}$$

Évidemment, `nb_iter` représente une borne maximale, car on peut interrompre l'itération dès qu'on a quitté le disque de rayon 2.

Cependant, dans un premier temps, on effectuera toujours `nb_iter` tests tout en enregistrant la valeur à laquelle on quitte le disque de rayon 2 (c'est cette valeur qu'on affichera). La tâche à réaliser est relativement simple, car aucun calcul ne nécessite des communications. Chaque thread calcule successivement des itérés

sur sa propre partie du carré [1]. Pour cette raison, ce genre de problèmes est appelé trivialement parallèle. On profitera donc de cette simplicité pour écrire un code bien structuré. Dans un second temps, vous devrez réfléchir à un moyen d'équilibrer la charge de chaque thread de façon statique. Dans ce cas-là, on ira donc pas jusqu'à `nb_iter` mais chaque boucle s'arrêtera au moment où on sort du disque de rayon 2.

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.gz` [2]). Affichez l'itération, k, à laquelle $z_k > 2$ en niveaux de gris (si vous avez du temps à perdre vous pouvez construire vos propres colormaps, chaque canal de couleur a la même intensité). Faites gérer l'affichage par un nouveau thread. Vous pouvez aller sur la page https://bit.ly/2HdQ6Jb pour trouver des exemples de valeurs de c et les résultats attendus.

Comparez les performances obtenues entre votre code tournant sur un seul thread avec un code tournant sur plusieurs threads. Faites également la comparaison entre la version avec l'équilibrage de charge et sans. Finalement, comparez la performance de votre code multi-threadé exécuté sur un seul thread avec un code purement séquentiel. Vous pouvez même si cela vous amuse tracer un graphe de la performance en fonction du nombre de threads utilisés.

Comme travail supplémentaire, affichez une animation en faisant varier c de façon continue. Une jolie animation peut être obtenue en prenant

$$c = 0.7885e^{ia}, \tag{7}$$

où $a \in [0, 2\pi[$.

5 Conseils

Avant d'écrire le code multi-threadé écrivez un code séquentiel. Une fois que vous avez confiance en votre code séquentiel il vous servira de référence pour déterminer si votre code concurrent fonctionne comme il faut. En effet, les deux codes doivent donner **exactement** les mêmes résultats.

Afin de vous assurer du bon fonctionnement de vote programme multi-threadé, il faut le lancer un grand nombre de fois et qu'à chaque fois le résultat soit le même. N'hésitez pas à écrire un petit script qui exécute votre code un grand nombre de fois, avec un nombre de threads variable et vérifiant que vos résultats sont toujours cohérents.

6 Corrigé

Je ne donnerai aucun corrigé à ces exercices. Vous pouvez si vous le souhaitez, me rendre votre code afin d'avoir un feedback. Si un · e étudiant · e (ou un groupe) souhaite soumettre sa version du code, je la relirai et la mettrai à disposition comme corrigé sur `cyberlearn` (avec un bonus sur une des notes du semestre).

1. Une façon simple (mais pas la seule) de séparer le carré est de le découper en bandes qui ont toutes la même largeur plus ou moins.
2. 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.