TD/TP 2 et 3 : Ensemble de Mandelbrot

Version du 20 août 2015

1 Introduction

L'ensemble de Mandelbrot est constitué des points c du plan complexe C pour lesquels le schéma itératif suivant :

$$\begin{cases} z_0 = 0 \\ z_{n+1} = z_n^2 + c \end{cases} \quad (1)$$

ne diverge pas. En posant $z = x + iy$ et $c = a + ib$, l'équation (1) se réécrit :

$$\begin{cases} x_{n+1} = x_n^2 - y_n^2 + a \\ y_{n+1} = 2x_n y_n + b \end{cases}$$

avec les conditions initiales $x_0 = y_0 = 0$.

On peut montrer que s'il existe un entier n tel que $|z_n| \geq 2$ (soit $|z_n|^2 = (x_n^2 + y_n^2) \geq 4$), alors le schéma (1) diverge. Pour en savoir plus, on pourra consulter [1, 2].

2 Structure des images en mémoire

Une image est un tableau à deux dimensions. Chaque élément de ce tableau s'appelle un *pixel*, abbréviation de *picture element*. La valeur de ce point est, selon le type d'image, une valeur de niveau de gris, une couleur ou une valeur de radiance. Ce tableau est organisé en mémoire ligne par ligne : on trouve la première ligne, puis la seconde, *etc*. En particulier, nous manipulerons des images codées sur un octet (taille d'un pixel), et la valeur de chaque pixel (donc un entier compris entre 0 et 255) représente un index dans une table de couleur.

En C, cela se traduit par :

```
{
  unsigned char *Image;
  int dimx, dimy;         // largeur et hauteur de l'image
  int i,j;               // indices pour parcourir les pixels de l'image

  Image = (unsigned char *) malloc (sizeof(unsigned char)*dimx*dimy);

  for(i=0; i<dimy; i++)
    for(j=0; j<dimx; j++)
      Image[j+i*dimx] = 0;  // acces au pixel i,j
}
```

Par convention, le pixel de coordonnées $(0, 0)$ est le point supérieur gauche d'une image affiché à l'écran, et c'est donc aussi le premier élément du tableau `Image` en mémoire.

3 Format d'image

Il existe une très grande multitude de format d'image, c'est-à-dire de façon de stocker dans un fichier une image. Nous utilisons le format Sun Rasterfile qui a le grand mérite d'être très simple à mettre en œuvre et qui est visualisable avec la plupart des programmes d'affichage d'image [1].

Un fichier rasterfile est constitué d'une entête qui décrit les caractéristiques de l'image (taille, codage, ...), suivie d'une suite de mots de 1 octet décrivant la table des couleurs (si besoin). Ensuite vient l'image proprement dite, stockée sous la forme d'un tableau de données brutes.

1. par exemple `display` (de la suite logicielle `ImageMagick`) sur Linux

4 Algorithme séquentiel

Synopsis de l'algorithme :

1. Pour chaque point du domaine :
   (a) calculer le nombre d'itérations pour lequel le schéma itératif diverge
   (nombre maximal d'itérations limité à une «profondeur» fixée par l'utilisateur) ;
   (b) mettre à jour la valeur du pixel correspondant suivant :
   Si profondeur atteinte : couleur_pixel ← 255
   Sinon : couleur_pixel ← NombreIterations % 255
2. Sauver l'image.

5 Questions

1. Discuter de la manière dont l'algorithme présenté en section 4 pourrait être parallélisé.
2. Proposer une version parallèle de cet algorithme qui répartisse de façon équilibrée les données à calculer sur chaque processeur. On utilisera les transmissions basiques de MPI (`MPI_Send` et `MPI_Recv`). En écrire le code C.
3. Comparer les temps d'exécution pour chaque processeur avec différents nombres de processeurs, ainsi que les efficacités parallèles obtenues. On étudiera en particulier le cas à 8 processeurs avec les paramètres par défaut.
4. L'équilibrage de charge précédent possède un gros défaut. Lequel ? Proposer un autre équilibrage de charge pour résoudre ce problème.
5. Implémenter en C l'algorithme correspondant à ce nouvel équilibrage de charge.

Références

[1] The Fractal Geometry of the Mandelbrot Set, *Robert L. Devaney* `http://math.bu.edu/DYSYS/FRACGEOM/`

[2] The Spanky Fractal Database, *Noël Giffin*, `http://spanky.triumf.ca/www/welcome1.html`

