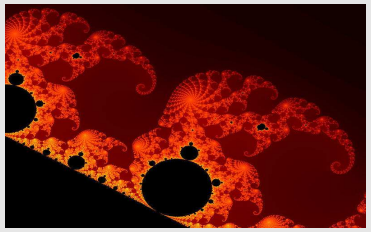
TP : Ensemble de Mandelbrot

LE205 - 2011-2012

Objectifs du TP :

L'objectif de ce TP est de réaliser des représentations graphiques de l'ensemble de Mandelbrot

Définitions :



Introduction
L'ensemble de Mandelbrot est un ensemble fractal découvert au début du vingtième siècle par les mathématiciens Gaston Julia et Pierre Fatou. Benoît Mandelbrot en a proposé une visualisation au début des années 1980. Le nom et la définition actuelle sont dus à Adrien Douady.

Préambule
Un nombre complexe *c* peut être représenté comme un point dans le plan en associant sa partie réelle à son abscisse et sa partie imaginaire à son ordonnée. Ainsi, la commande Octave/Matlab pour afficher un point *c* (ou plusieurs) est :
plot(real(c), imag(c))

Définition
L'ensemble de Mandelbrot est une fractale définie comme l'ensemble des points *c* du plan (voir préambule) pour lesquels la suite définie par récurrence par :

$$z_{n+1} = z_n^2 + c$$

avec $z_0 = 0$
ne tend pas vers l'infini (en module).

Propriété utile
S'il existe un entier N tel que $|z_N| > 2$, alors la suite diverge.

Représentation graphique
En pratique on fixe un nombre maximum *M* d'itérations. Si le module ne dépasse pas la valeur *2* après *M* itérations, on considère que la suite est convergente et que le point *c* appartient à l'ensemble de Mandelbrot.
Pour des raisons purement esthétiques, l'ensemble de Mandelbrot est souvent représenté avec des couleurs. Une couleur (le noir dans l'exemple ci-dessus) est attribué aux points appartenant à l'ensemble. Ensuite, la couleur dépend du nombre d'itérations nécessaires pour que la suite soit considérée divergente (En pratique, il s'agit du nombre d'itérations nécessaires pour que le module de *z* dépasse *2*).

Enoncé :

1) Créer la fonction `converge` qui prend en entrée un vecteur complexe *c* et l'entier *M* représentant le nombre maximum d'itération à tester. Cette fonction renvoie en sortie un vecteur contenant pour chaque élément du vecteur *c* le nombre d'itérations avant lequel la suite a été déterminée divergente. On peut envisager que le vecteur contienne un nombre particulier pour les cas où la suite n'a pas été déterminée divergente (on suppose alors qu'elle est convergente).

2) Créer le script `affiche_mandelbrot` qui fait l'affichage (en couleur) de l'ensemble de Mandelbrot complet (faire quelques recherches pour trouver les limites des axes adéquates). Trouver une palette de couleur, un nombre *M* et un résolution satisfaisante de façon à ce que la forme générale soit visible mais que l'affichage ne soit pas trop long. Essayer plusieurs valeurs de *M* pour voir les différences. Comment les expliquez-vous ?

3) Faire des affichages de quelques zones particulières de l'ensemble, essayer de trouver des formes différentes, mais aussi essayer de trouver des répliques plus petites de l'ensemble de départ qui justifient le caractère fractal de l'ensemble. Pour cela, il vous faudra changer les bornes des axes ainsi que la résolution.

4) Modifier le script `affiche_mandelbrot` de façon à ce qu'il demande à l'utilisateur un point et qu'il effectue un zoom autour de ce point (l'intensité du zoom pourra aussi être réglé par l'utilisateur). Pour cela, vous pourrez utiliser la fonction `input` de Octave/Matlab.

5) Faire une animation en choisissant un point fixe au centre de l'image et en zoomant progressivement autour de ce point. L'animation sera faite par une succession d'images à des zooms différents.

Sujet 2 : Automate cellulaire (jeu de la vie)

LE205 - 2011-2012

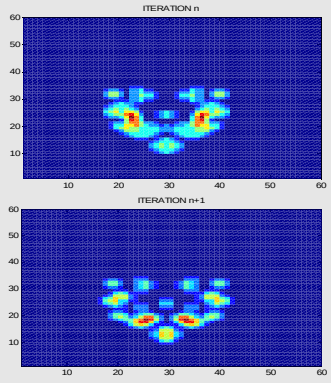
Objectifs du TP :

L'objectif de ce TP est de réaliser un automate cellulaire connu sous le nom de jeu de la vie

Définitions :

Introduction

(Wikipedia): Un **automate cellulaire** consiste en une grille régulière de «cellules» contenant chacune un «état» choisi parmi un ensemble fini et qui peut évoluer au cours du temps. L'état d'une cellule à l'itération N+*1* est fonction de l'état à l'itération N d'un nombre fini de cellules appelé son «voisinage». À chaque nouveau pas de temps, les mêmes règles sont appliquées simultanément à toutes les cellules de la grille, produisant une nouvelle «génération» de cellules dépendant entièrement de la génération précédente.



Règle du jeu de la vie

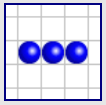
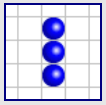
Le jeu de la vie est un automate cellulaire imaginé par John Horton Conway en 1970 qui malgré des règles très simples permet le développement de motifs extrêmement complexes. Elles sont le suivantes

On définit l'état vivant ou mort d'une cellule par la valeur 1 ou 0.

À chaque étape, l'évolution d'une cellule est entièrement déterminée par l'état de ses huit voisines de la façon suivante:

- Une cellule morte possédant exactement trois voisines vivantes devient vivante (elle naît).
- Une cellule vivante possédant deux ou trois voisines vivantes le reste, sinon elle meurt.

- Exemple de motif périodique:

La configuration  donne au tour suivant la configuration  qui redonne ensuite la première.

Enoncé:

1) Créer la fonction «evol_cel» qui prend en entrée une matrice Ma qui stocke l'état (0 ou 1) des cellules sur une grille à l'iteration N et renvoit une matrice Mb qui stocke l'état des cellules à l'iteration N+1.
Pour déterminer le nombre de voisins de chaque cellule qui détermine l'état de la cellule à l'itération suivante vous avez trois possibilités:
-Calculer à l'aide de boucles le nombre de cellules voisines vivantes (etat 1) de chaque cellule de la grille, cette solution la plus évidente est couteuse en temps de calcul.
- Calculer la somme de la matrice Ma avec sa translatée d'une cellule dans 8 directions différentes (haut, bas droite, gauche, diag bas droit, diag bas gauche, diag haut droit, diag haut gauche.
-Calculer le produit de convolution à 2D de la matrice Ma avec une matrice 3x3 de 1.
Sous matlab/octave le produit de convolution à 2D se calcule à l'aide de la fonction «conv2».
Dans tous les cas vous aurez besoin de la fonction «find» (cf aide sur la commande find en fin d'énnoncé).

2) Créer le script `affiche_cell` qui
- initialise une matrice de cellules avec des 1 et des 0, le choix est arbitraire mais vous pouvez par exemple générer de façon aléatoire des 1 et des 0 sur une matrice en combinant la fonction rand et une condition logique. Par exemple pour une matrice M, A=double(M>0.5), renvoie une matrice A de même taille que M avec des 1 ou des 0 suivant que la condition >0.5 est vérifiée ou non pour chaque élément de M. La commande double permet de convertir le type booléen en type double précision (type par défaut).
-Au sein d'une boucle sur un nombre max d'itérations Nmax, fait appel à la fonction evol_cel pour calculer la nouvelle matrice de cellules à chaque itérations et affiche la matrice sur un graphique 2D.

3) Si vous avez terminé les étapes 2 et 3 vous pouvez améliorer la fonction evol_cel afin de prendre en compte la durée de vie d'une cellule. C'est à dire qu'une cellule qui survit d'une itération à l'autre verra sa valeur augmenter de +1, ainsi une cellule qui nait à l'itération 1 et survit jusqu'à itération 10 aura la valeur 10. A l'affichage (avec pcolor par exemple) la couleur indiquera alors l'age de la cellule.
Attention pour que l'algorithme du jeu de la vie continue de fonctionner correctement vous aurez besoin de réinitialiser dans une matrice temporaire la valeur de toutes le cellules à l'état vivant (1) ou mort (0) à chaque itération avant d'appliquer la règle...

4) Vous pouvez trouver sur internet de nombreux exemples de motifs initiaux relativement simples qui donnent ensuite des structures complexes et/ou périodiques. Proposer à l'utilisateur de choisir entre quelques uns de ces motifs (aidez vous des fonctions « input » et « menu »)

Aide commande find

`I = FIND(X) renvoie les indices linéaires correspondant aux éléments non nuls de la matrice X.`

L'indice linéaire est compté comme i=(c-1)*L+l, avec c=no colonne de l'élement , l=no ligne de l'élement et L=le nombre de lignes de la matrice. L'élément x_{32} à par exemple pour indice linéaire i=(2-1)*3+2=5.

Par exemple pour la matrice A =[0 0 2
1 0 0
0 0 0]

A est de taille (LxC)=(3x3) (L=3, C=3)

i=find(A) renvoie i=[2, 7]. En effet on a une valeur non nulle à la ligne l=2 colonne c=1=> indice linaire i=(1-1)*3+2=2, et à la ligne 1 colonne 3 indice linaire i=(3-1)*3+1=7.

Il est possible d'ajouter une condition à la fonction find par exemple

i=find(A>1.5) ne renvoie que l'indice linéaire i=7 correspondant à l'élément a_{13}=2 (>1.5).

On peut utiliser les indices i renvoyés par la commande find pour modifier les valeurs d'une matrice par exemple la suite de commandes:

i=find(A); A(i)=5 renvoie

A =[0 0 5
5 0 0
0 0 0]