Informatique

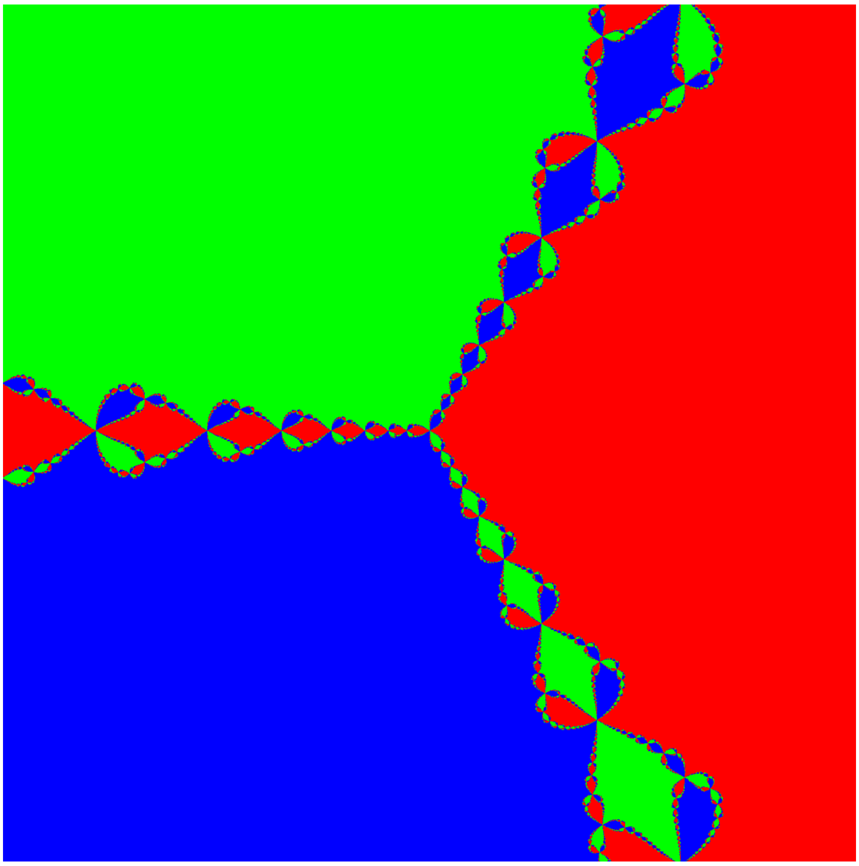
7
Matrices de pixels et images

TD 7-8

Fractales de Newton

Méthode de Newton

Fractale de Newton



Soit un nombre complexe $z_0 = x + iy$.
Soit la fonction $f(z) = z^3 - 1$.
Soient a, b et c les trois solutions de l'équation $f(z) = 0$. Ce sont donc les racines cubiques de l'unité.

Les variables x et y doivent être contenues dans les intervalles suivants :

$$x \in [-1{,}5; 1{,}5]$$
$$y \in [-1{,}5; 1{,}5]$$

On admettra que la méthode de Newton pour résoudre l'équation $f(z) = 0$ s'étend aux complexes.
On prendre comme critère $|f(z)| < \varepsilon$ avec $\varepsilon = 10^{-6}$

Notre objectif est d'obtenir une image BMP dans laquelle à chaque pixel est associée une couleur correspondant soit :

- Bleu : convergence vers la solution a
- Vert : convergence vers la solution b
- Rouge : convergence vers la solution c
- Blanc : éventuelle non convergence

Chargement du code élèves

Le code élève est en lien ici : LIEN CODE ELEVES

Ce code

- Crée une image noire de dimensions :
    - Nombre de colonnes : `Nb_Colonnes` – Valeur que vous pourrez modifier par la suite pour obtenir une belle image
    - Nombre de lignes : Nombre proportionnel à `Nb_Colonnes` respectant les proportions du domaine de définition ($x \in [-1{,}5; 1{,}5]$, $y \in [-1{,}5; 1{,}5]$), soit des proportions 1/1
- Affiche l'image créée

Il est très simple de modifier le triplet RGB d'un pixel en écrivant :

```
Image[l_pix,c_pix] = [255,255,255]
```

Le pixel à la ligne `l_pix` et à la colonne `c_pix` se retrouve transformé en un pixel blanc.
Il nous reste donc à créer un code qui modifie le triplet RGB de chaque pixel de l'image en fonction de ses coordonnées et le la convergence de la méthode de Newton.

Question 1: Téléchargez le code proposé et vérifiez qu'une image noire est bien affichée lors de son exécution

Méthode de Newton

Question 2: Créer les fonctions f(z) et fp(z), dérivée exacte de f(z)

Question 3: Créer une fonction Newton(f,fp,z0,Crit) résolvant l'équation f(z)=0 dans le plan complexe par la méthode de Newton avec le critère Crit en partant du complexe z0 et avec fp la fonction dérivée de f. Cette fonction renverra, la solution et le nombre d'itérations réalisées.

Remarques

- On traitera le cas où fp s'annule, auquel cas on renverra la solution z actuelle et un nombre d'itérations très grand, par exemple 999999999 pour indiquer la non convergence.
- Le module de z s'écrit $abs(z)$

Question 4: Créer une fonction Convergence_Newton(x,y) qui renvoie la solution et le nombre d'itérations de la méthode de Newton en partant de z0=x+iy avec un critère de 10^{-6}

Rappel : Créer le complexe $x + iy$ s'écrit $c = complex(x, y)$

Vérifiez :

```
>>> Convergence_Newton(0,0)
(0j, 999999999)

>>> Convergence_Newton(-1,-1)
((-0.4999999999999555-0.8660254037846933j), 5)

>>> Convergence_Newton(-1,1)
((-0.4999999999999555+0.8660254037846933j), 5)

>>> Convergence_Newton(1,-1)
((0.9999999999999994+4.556244651765188e-16j), 8)

>>> Convergence_Newton(-1,-1)
((-0.4999999999999555-0.8660254037846933j), 5)
```

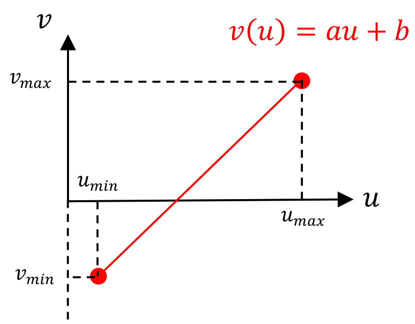
Gestion des coordonnées des pixels

Les pixels étant définis par la donnée d'une ligne et d'une colonne, leurs numéros de ligne et colonne sont des entiers allant de 0 à la valeur de (Nb_Lignes-1) ou (Nb_Colonnes-1). La suite, elle, ne converge que dans les intervalles suivants :

$$x \in [-1,5; 1,5]$$
$$y \in [-1,5; 1,5]$$

Nous allons donc faire en sorte d'associer à chaque pixel de l'image des coordonnées dans ces intervalles.

Dans un premier temps, proposons une fonction qui permet par une fonction affine d'adapter une échelle en pixels en une échelle en abscisses ou en ordonnées :

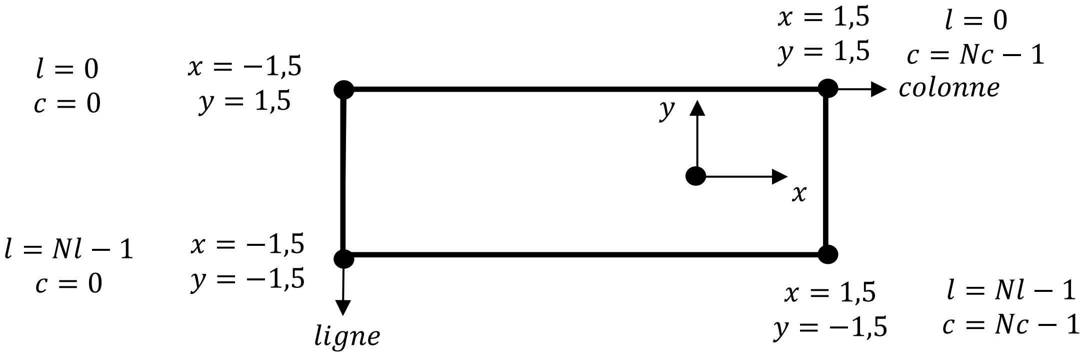


Question 5: Créer une fonction Echelle(u,u_min,u_max,v_min,v_max) qui renvoie le nombre v(u) comme proposé sur la figure ci-dessus

Vérifiez :

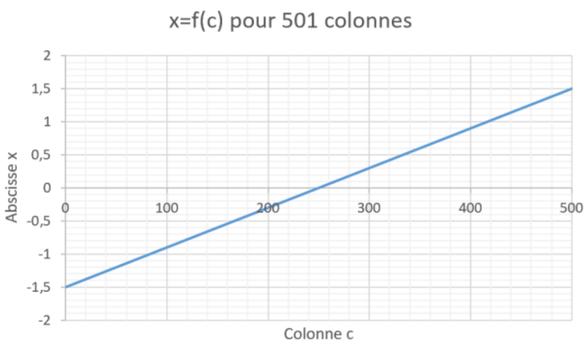
```
>>> Echelle(20,0,100,0,1)
0.2

>>> Echelle(20,-100,100,-10,10)
2.0
```

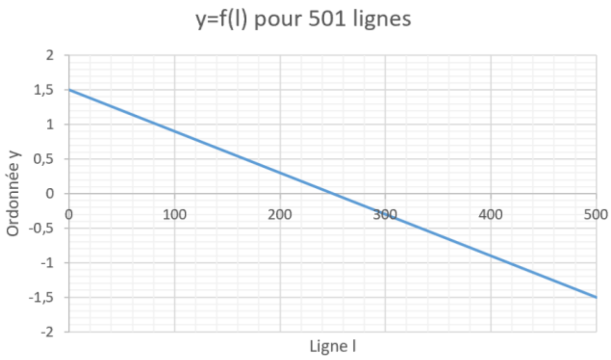
La figure ci-dessous présente la correspondance entre ligne et colonne d'un pixel et ses coordonnées dans les intervalles d'étude du domaine de Mandelbrot :



Pour déterminer les coordonnées d'un pixel de ligne l_pix et de colonne c_pix, il faut :

- Utiliser la fonction Echelle afin d'adapter la colonne c_pix dans $[0; Nc - 1]$ l'intervalle $[-1{,}5; 1{,}5]$ dans cet ordre
- Utiliser la fonction Echelle afin d'adapter la ligne l_pix dans $[0; Nl - 1]$ l'intervalle $[-1{,}5; 1{,}5]$ dans cet ordre





Question 6: Créer une fonction Coordonnees_Pixel(l_pix,c_pix) qui renvoie les coordonnées x et y associées à un pixel de ligne l_pix et de colonne c_pix

Remarques :

- Il faut évidemment que les coordonnées d'un pixel soient déterminées automatiquement en fonction des grandeurs Nb_Lignes et Nb_Colonnes
- Attention, lignes et colonnes évoluent dans les intervalles $[0, Nb\_Lignes - 1]$ et $[0, Nb\_Colonnes - 1]$

Vérifiez :

```
>>> Coordonnees_Pixel(0,0)
(-1.5, 1.5)

>>> Coordonnees_Pixel(Nb_Lignes-1,Nb_Colonnes-1)
(1.5, -1.5)

>>> Coordonnees_Pixel(Nb_Lignes-1,0)
(-1.5, -1.5)

>>> Coordonnees_Pixel(0,Nb_Colonnes-1)
(1.5, 1.5)
```

Création de la fractale de newton

Il reste maintenant à modifier chacun des pixels de l'image en fonction de ses coordonnées en déterminant vers quelle racine de l'unité la méthode de Newton converge depuis ce point initial.

Question 7: Créer une fonction Couleur_Pixel(l_pix,c_pix), qui calcule les coordonnées du pixel concerné, détermine la solution issue de la fonction Convergence_Newton et qui affecte une couleur au pixel concerné en fonction de la solution vers laquelle la méthode a convergé comme précisé plus haut

Remarques :

- On pourra au choix créer les racines de l'unité avec `complex(a,b)` ou `r*np.exp(i*teta)`
- On pourra ici aussi prendre un critère de 10^{-6} pour estimer la proximité à l'une des trois racines cubiques de l'unité.
- Les couleurs :
    - Rouge `[255,0,0]`
    - Vert `[0,255,0]`
    - Bleu `[0,0,255]`
    - Blanc `[255,255,255]` mais pas très utile, l'image est déjà blanche…

Question 8: Créer la fonction Fractale_Newton() qui parcourt tous les pixels de l'image et qui affecte la couleur correspondante

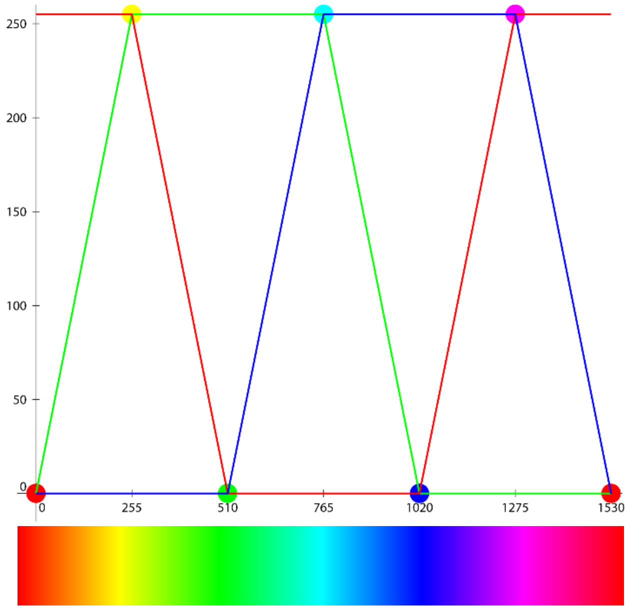
Question 9: Afficher le résultat pour différentes tailles d'images en jouant sur le paramètre Nb_Colonnes

Mise en place des couleurs de l’arc en ciel

Pour afficher les couleurs de l’arc en ciel, nous allons utiliser le nombre d’itérations nécessaires pour obtenir la convergence de la méthode de Newton.

Nous allons définir une fonction Arc_En_Ciel(N,N_max) qui associe au nombre N une couleur de l’arc en ciel sachant que l’on veut utiliser toute la plage de couleurs disponibles pour N allant de 0 à N_max.

Le principe d’obtention d’une couleur de l’arc en ciel est illustré ci-dessous. Pour une variable x dans l’intervalle [0,1530], on associe les 3 couleurs :



Source

Ce qui donne les formules suivantes :

$x \in [0{,}1530]$	$x \in [-\infty, \infty]$
$R = \left\{\begin{array}{c} 0 \le x \le 255 \\ 255 \le x \le 510 \\ 510 \le x \le 1020 \\ 1020 \le x \le 1275 \\ 1275 \le x \le 1530 \end{array}\right. \Rightarrow \begin{array}{c} 255 \\ 510 - x \\ 0 \\ x - 1020 \\ 255 \end{array}$	$R = \left\{\begin{array}{c} x \le 255 \\ 255 \le x \le 510 \\ 510 \le x \le 1020 \\ 1020 \le x \le 1275 \\ 1275 \le x \end{array}\right. \Rightarrow \begin{array}{c} 255 \\ 510 - x \\ 0 \\ x - 1020 \\ 255 \end{array}$
$G = \left\{\begin{array}{c} 0 \le x \le 255 \\ 255 \le x \le 765 \\ 765 \le x \le 1020 \\ 1020 \le x \le 1530 \end{array}\right. \Rightarrow \begin{array}{c} x \\ 255 \\ 1020 - x \\ 0 \end{array}$	$G = \left\{\begin{array}{c} x \le 255 \\ 255 \le x \le 765 \\ 765 \le x \le 1020 \\ 1020 \le x \end{array}\right. \Rightarrow \begin{array}{c} \max(x,0) \\ 255 \\ 1020 - x \\ 0 \end{array}$
$B = \left\{\begin{array}{c} 0 \le x \le 510 \\ 510 \le x \le 765 \\ 765 \le x \le 1275 \\ 1275 \le x \le 1530 \end{array}\right. \Rightarrow \begin{array}{c} 0 \\ x - 510 \\ 255 \\ 1530 - x \end{array}$	$B = \left\{\begin{array}{c} x \le 510 \\ 510 \le x \le 765 \\ 765 \le x \le 1275 \\ 1275 \le x \end{array}\right. \Rightarrow \begin{array}{c} 0 \\ x - 510 \\ 255 \\ \max(1530 - x, 0) \end{array}$

Question 10: Mettre en place la fonction Arc_En_Ciel(N,N_max) qui utilise la fonction Echelle pour adapter le nombre N à l'intervalle proposé ci-dessus et qui renvoie le triplet [R,G,B] associé – On renverra la couleur en 0 si N<0 et la couleur en N_Max si N>N_Max

Vérifiez :

```
>>> Arc_En_Ciel(-50,100)
[255, 0, 0]

>>> Arc_En_Ciel(0,100)
[255, 0, 0]

>>> Arc_En_Ciel(25,100)
[128, 255, 0]

>>> Arc_En_Ciel(50,100)
[0, 255, 255]

>>> Arc_En_Ciel(75,100)
[127, 0, 255]

>>> Arc_En_Ciel(100,100)
[255, 0, 0]

>>> Arc_En_Ciel(150,100)
[255, 0, 0]
```

Question 11: Modifier votre fonction Couleur_Pixel afin de prendre en compte cette couleur en appelant l'arc en ciel pour N le nombre d'itérations de convergence et N_Max = 100

Question 12: Afficher le résultat pour différentes tailles d'images