Informatique

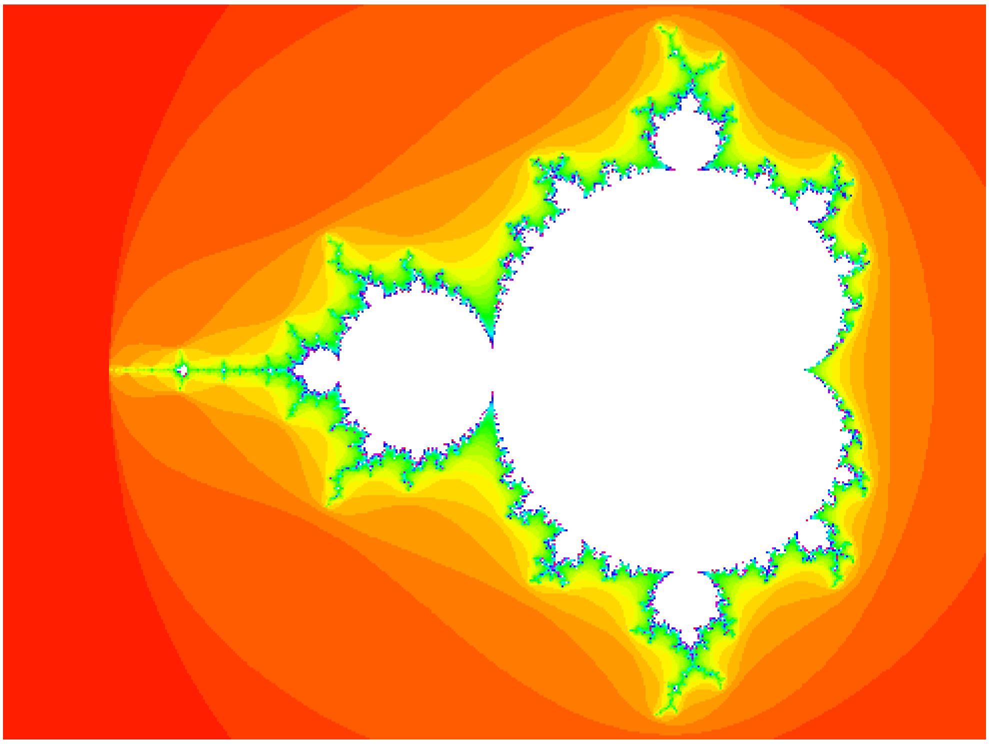
7
Matrices de pixels et images

TD 7-7

Fractale de Mandelbrot

Récursivité

Fractale de Mandelbrot



L'ensemble de Mandelbrot est défini comme l'ensemble des nombres $c = x + iy$ du plan complexe pour lesquels la suite définie par récurrence ci-dessous ne tend pas vers l'infini :

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

On pourra en réalité se limiter à vérifier que la suite ne dépasse pas la valeur de 3 pour $n_{max} = 50$.

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

$$x \in [-2{,}35; 0{,}85]$$
$$y \in [-1{,}2; 1{,}2]$$

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

- Blanc : La suite ne diverge pas
- Couleurs de l'arc en ciel : La suite diverge de plus en plus vite du bleu verts le rouge

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 de la suite de Mandelbrot ($x \in [-2{,}35; 0{,}85]$, $y \in [-1{,}2; 1{,}2]$), soit des proportions 3.2/2.4
- 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 suite de Mandelbrot.

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

Suite de Mandelbrot

Lorsque la suite de Mandelbrot diverge, elle peut diverger très rapidement et peut atteindre des valeurs non représentables par le système d'exploitation après moins de **50 itérations** :

```
z = z**2 + c
OverflowError: complex exponentiation
```

Comme nous allons nous arrêter dès que la suite dépasse un **module de 3**, je vous propose la modification suivante pour éviter ce problème :

$$\begin{cases} z_0 = 0 \\ z_{n+1} = \begin{cases} ({z_n}^2 + c\ )\ si\ |{z_n}^2 + c| < 3 \\ 3\ sinon \end{cases} \end{cases}$$

Ainsi, dès que la divergence sera atteinte, z_n sera toujours égale à 3.

Question 2: Créer une fonction récursive Suite_Mandelbrot(n,c,z0) où c est un complexe quelconque et qui renvoie le résultat tel que proposé ci-dessus.

Rappels :

- Le module de z s'écrit $abs(z)$
- Pour créer un complexe : `a = 1+1j*2` ou `a = complex(1,2)`

Remarques :

- Attention aux indices dans votre fonction, la définition est entre n+1 et n, vous risquez de programmer avec n et n-1 alors ne vous trompez pas…
- EVITER à tout prix d'appeler deux fois la même fonction récursive… Lors de l'évaluation de ${z_n}^2 + c$, histoire de ne pas avoir une complexité en temps gigantesque pour de petites images

Vérifiez :

```
>>> Suite_Mandelbrot(50,0.1,0)
0.11270166537925831

>>> Suite_Mandelbrot(50,0.3,0)
3

>>> Suite_Mandelbrot(50,complex(0.1,0.1),0)
(0.09362728698078197+0.12303975734127459j)
```

Pour info, quand on ne limite pas la valeur, on a :

```
>>> Suite_Mandelbrot(50,0.255,0)
3.6097536965368177e+78
```

. La divergence est rapide…

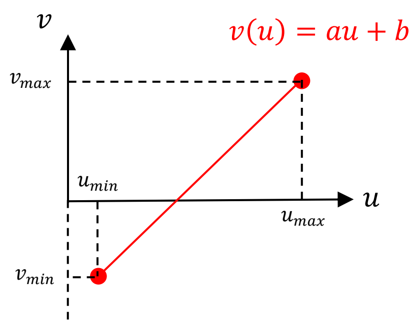
Gestion des coordonnées des pixels

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 :

$$x \in [-2{,}35; 0{,}85]$$
$$y \in [-1{,}2; 1{,}2]$$

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 :

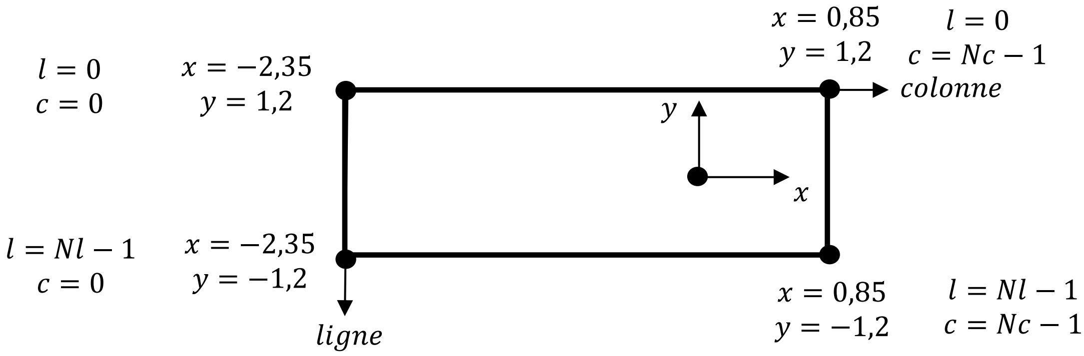


Question 3: Créer une fonction Echelle(u,u_min,u_max,v_min,v_max) qui renvoie le nombre v(u) comme proposé sur la figure ci-dessus

Vérifiez :

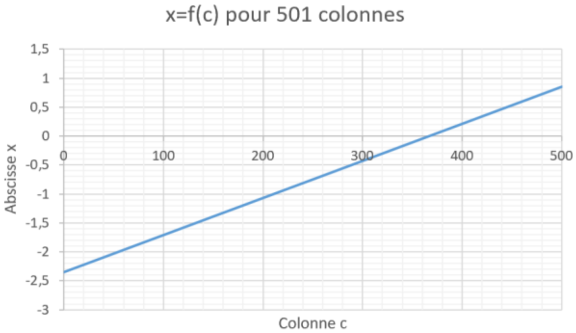
```
>>> Echelle(20,0,100,0,1)
0.2

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

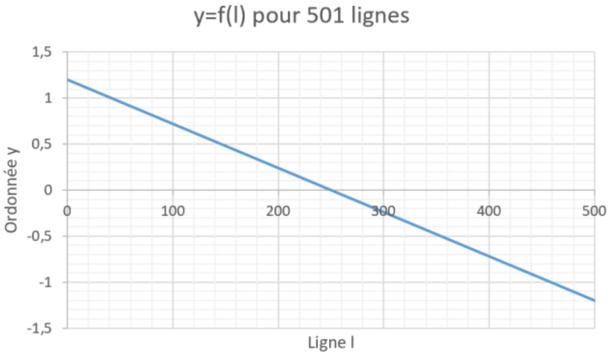
La figure ci-dessous présente la correspondance entre ligne et colonne d'un pixel et ses coordonnées dans les intervalles d'étude du domaine de Mandelbrot :



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 $[-2{,}35; 0{,}85]$ dans cet ordre
- Utiliser la fonction Echelle afin d'adapter la ligne l_pix dans $[0; Nl - 1]$ l'intervalle $[1{,}2; -1{,}2]$ dans cet ordre





Question 4: 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)
(-2.35, 1.2)

>>> Coordonnees_Pixel(Nb_Lignes-1,0)
(-2.35, -1.2)

>>> Coordonnees_Pixel(0,Nb_Colonnes-1)
(0.8500000000000001, 1.2)

>>> Coordonnees_Pixel(Nb_Lignes-1,Nb_Colonnes-1)
(0.8500000000000001, -1.2)
```

Création de la fractale en noir et blanc

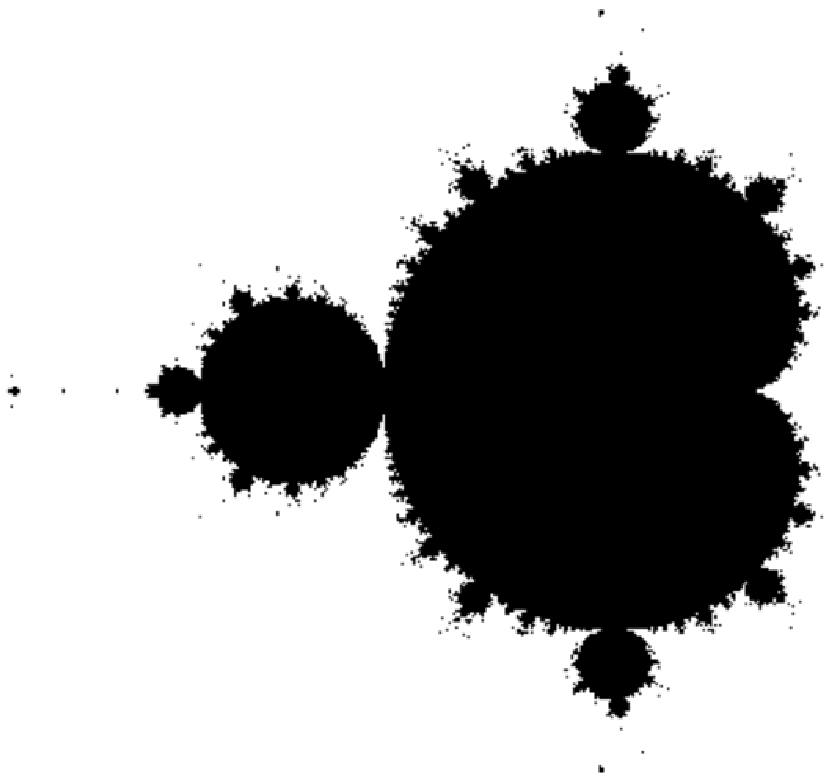
Il reste maintenant à modifier chacun des pixels de l'image en fonction de ses coordonnées en vérifiant si la suite converge ou diverge en ce point.
On rappelle que l'on programmera le calcul pour n = 50.

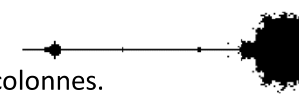
Question 5: Créer une fonction Couleur_Pixel(l_pix,c_pix), qui calcule les coordonnées du pixel concerné, détermine si la suite de Mandelbrot converge en ce point, et affecte au pixel concerné la couleur blanche (R=G=B=255) si la suite diverge, et noire (R=G=B=0) sinon

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

Question 6: Créer la fonction Fractale_Mandelbrot() qui parcourt tous les pixels de l'image et qui affecte la couleur correspondante

Remarque : on pourra calculer et afficher un compteur en % indiquant l'avancée du calcul en utilisant un pas dépendant de la ligne l et la colonne c traitées et un pourcentage utilisant le nombre total de pas Nb_Lignes*Nb_Colonnes (ne pas oublier de multiplier par 100). ATTENTION : la présence du print ralenti beaucoup l'exécution…



Si vous voyez une ligne noire, ce n'est pas grave, elle apparait par exemple à 1000 colonnes. 

Question 7: 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, savoir si oui ou non la suite à convergé ne suffit plus. Il faut en effet savoir au bout de combien d'itérations la suite à divergé.

Question 8: Modifiez votre fonction Suite_Mandelbrot pour qu'elle renvoie à chaque itération un second terme correspondant à la dernière valeur de n où le calcul n'a pas divergé (module inférieur à 3)

Vérifiez :

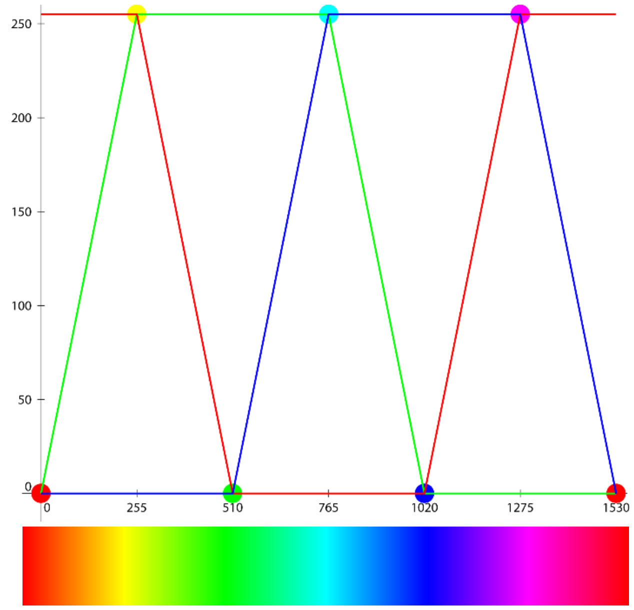
```
>>> Suite_Mandelbrot(50,0.1,0)
(0.11270166537925831, 50)

>>> Suite_Mandelbrot(50,0.2,0)
(0.2763932022500072, 50)

>>> Suite_Mandelbrot(50,0.3,0)
(3, 12)
```

Question 9: Adaptez votre fonction Couleur_Pixel pour qu'elle tienne compte de ce renvoie supplémentaire du nombre d'itérations avant divergence

A ce stade, vous avez le même résultat que dans la première partie, à la différence que vous avez accès en plus, à la valeur de n à la dernière convergence de la suite.

Il ne reste plus qu'à 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

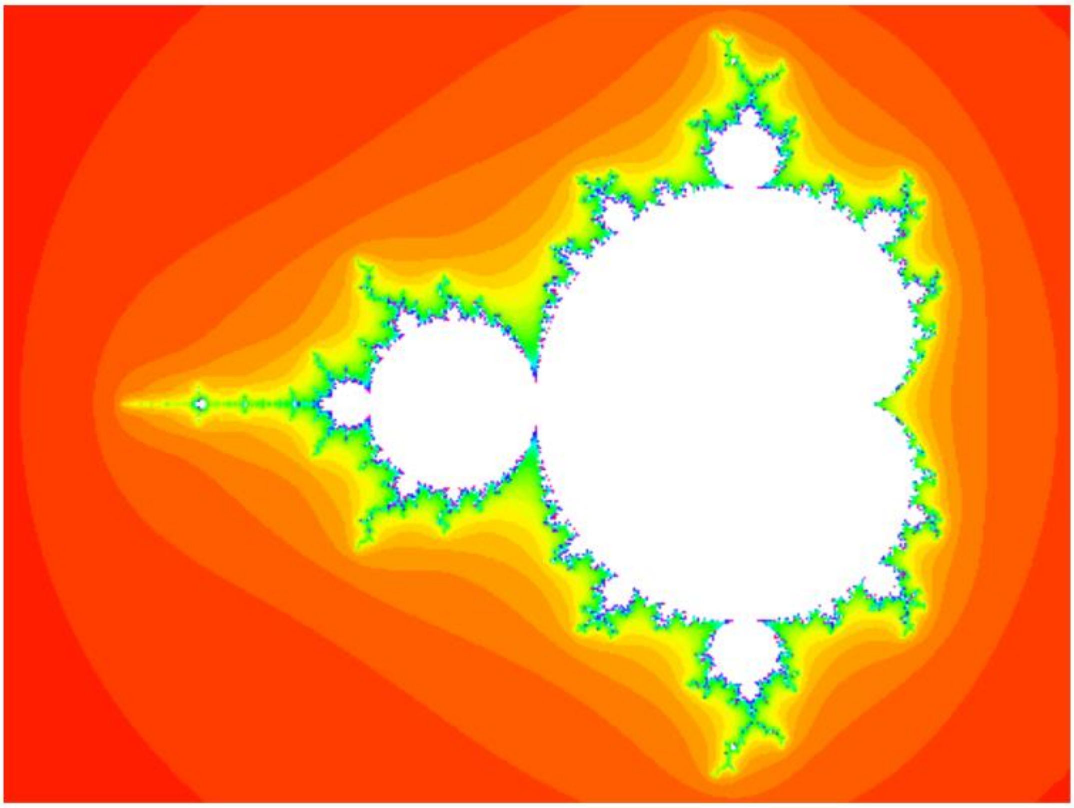
Ce qui donne les formules suivantes :

$x \in [0,1530]$	$x \in [-\infty, \infty]$
$R = \begin{cases} 0 \leq x \leq 255 & 255 \\ 255 \leq x \leq 510 & 510 - x \\ 510 \leq x \leq 1020 \Rightarrow & 0 \\ 1020 \leq x \leq 1275 & x - 1020 \\ 1275 \leq x \leq 1530 & 255 \end{cases}$ $G = \begin{cases} 0 \leq x \leq 255 & x \\ 255 \leq x \leq 765 \Rightarrow & 255 \\ 765 \leq x \leq 1020 & 1020 - x \\ 1020 \leq x \leq 1530 & 0 \end{cases}$ $B = \begin{cases} 0 \leq x \leq 510 & 0 \\ 510 \leq x \leq 765 \Rightarrow & x - 510 \\ 765 \leq x \leq 1275 & 255 \\ 1275 \leq x \leq 1530 & 1530 - x \end{cases}$	$R = \begin{cases} x \leq 255 & 255 \\ 255 \leq x \leq 510 & 510 - x \\ 510 \leq x \leq 1020 \Rightarrow & 0 \\ 1020 \leq x \leq 1275 & x - 1020 \\ 1275 \leq x & 255 \end{cases}$ $G = \begin{cases} x \leq 255 & \max(x, 0) \\ 255 \leq x \leq 765 \Rightarrow & 255 \\ 765 \leq x \leq 1020 & 1020 - x \\ 1020 \leq x & 0 \end{cases}$ $B = \begin{cases} x \leq 510 & 0 \\ 510 \leq x \leq 765 \Rightarrow & x - 510 \\ 765 \leq x \leq 1275 & 255 \\ 1275 \leq x & \max(1530 - x, 0) \end{cases}$

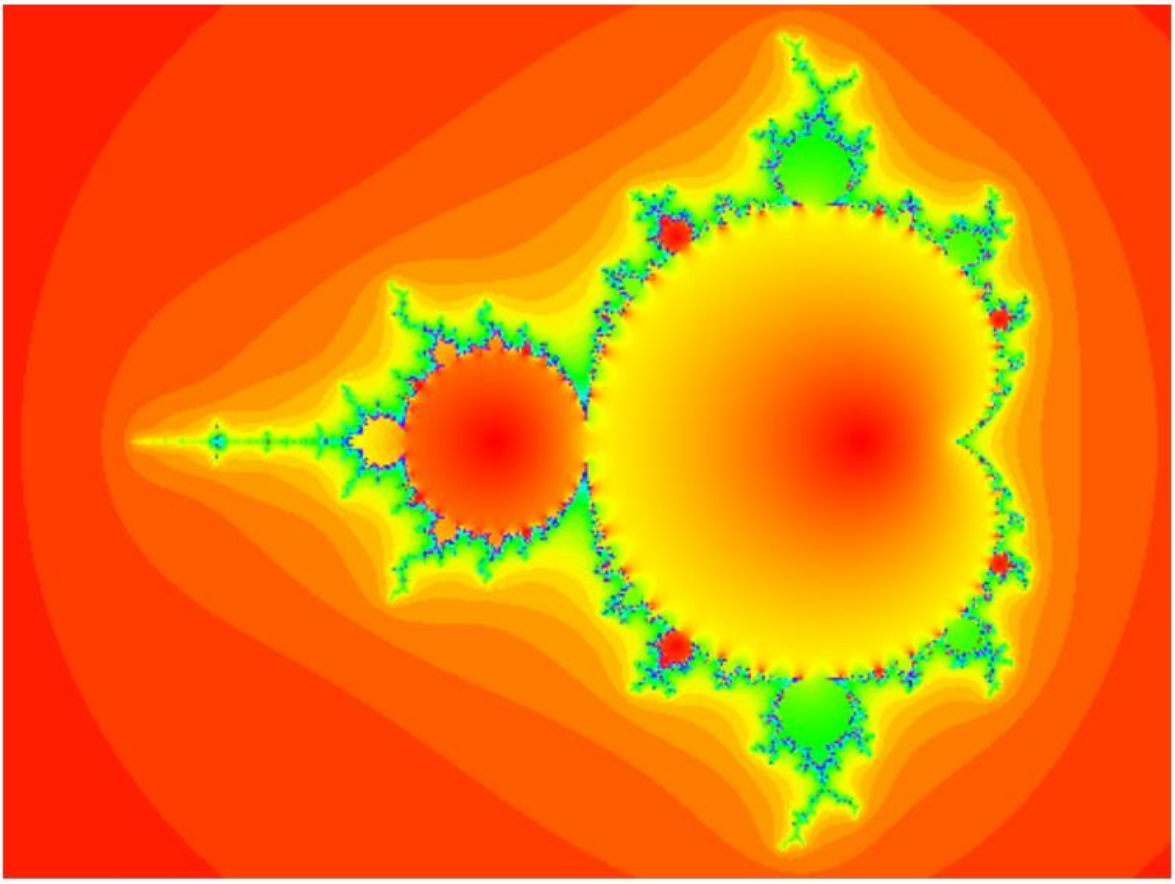
Je vous propose dans le code élèves la fonction Arc_En_Ciel(N,N_max) qui utilise la fonction Echelle pour adapter le nombre N à l'intervalle proposé ci-dessus et qui renvoie le triplet [R,G,B] associé – Elle renvoie la couleur en 0 si N<0 et la couleur en N_Max si N>N_Max.

Question 10: Modifier votre fonction Couleur_Pixel afin de prendre en compte cette couleur lorsque la suite diverge

Question 11: Afficher le résultat pour différentes tailles d'images



Encore un peu de temps ?

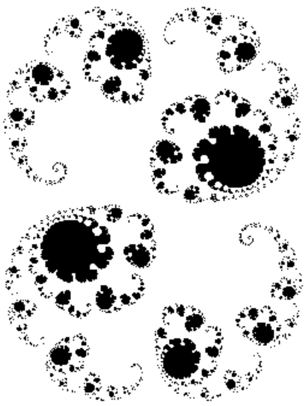
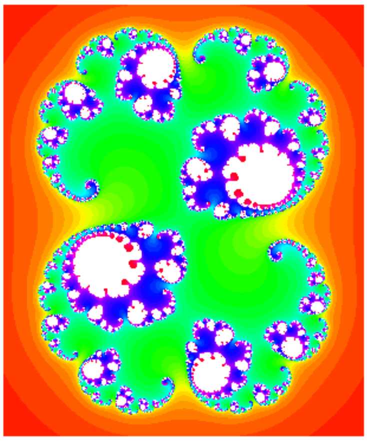
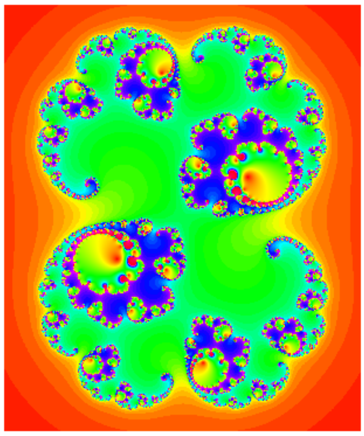


Question 12: Adaptez votre code afin d'ajouter les couleurs de l'arc en ciel à l'intérieur du domaine de convergence en fonction de la valeur du module de z_{50}, inférieur à 3…

Remarque : Les fonction Suite_Mandelbrot peut renvoyer un nombre supérieur à 3 si à l'étape 49, le module était inférieur à 3. On veillera donc à imposer un module de 3 si le module renvoyé dépasse 2 afin que la fonction Arc_En_Ciel fonctionne dans l'intervalle [0,3]

Ensemble de Julia

Il vous est maintenant possible de tracer l'ensemble de Julia :

Pour cela, au lieu de fixer z_0 et de tracer la convergence de la suite pour différents c, il suffit de fixer c et de tracer la convergence de la suite pour différents z_0.
Je vous suggère de prendre $c = 0{,}285 + 0{,}01\mathrm{i}$ et de redéfinir le Xmin et Xmax de l'image :

```
X_Min = -1
X_Max = 1
```