Tracé vectoriel de fractales

Matthijs Douze
IRIT/ENSEEIHT — douze@enseeiht.fr

Résumé

Les fractales sont des figures géométriques dans lesquelles une « forme » caractéristique est présente à plusieurs résolutions différentes. Nous nous intéressons ici à un type de fractales particulières : les fractales à temps d'échappement*. L'approche classique employée pour visualiser celles-ci consiste à les échantillonner sur une grille régulière. Nous explorons plusieurs manières alternatives pour les représenter, et la manière dont nous les avons implantées.*

Mots Clef

Fractales, images vectorielles.

Introduction

La notion mathématique de fractale a été introduite par Benoît Mandelbrot [1]. Il avait remarqué que dans certains phénomènes d'apparence complexes, des caractéristiques locales se retrouvaient à des endroits divers *et* à des échelles différentes.
Les fractales s'inscrivent dans une hiérarchie de structures géométriques (figure 1).
Les fractales sont présentes dans la nature, et utilisées partout. En particulier, le système de numération arabe est fractal : on code de la même manière (au moyen de chiffres) les détails à grande et à petite échelle (figure 2).
Mandelbrot a mis en évidence le comportement intéressant d'une classe de fractales appelées *escape-time fractals*.

1 Fractales à temps d'échappement

Une fractale à temps d'échappement est définie par une fonction complexe f et un critère N.

$$\begin{array}{rccc} f: & \mathbb{C}\times\mathbb{C} & \longrightarrow & \mathbb{C} \\ N: & \mathbb{C} & \longrightarrow & \mathbb{R}^+ \end{array}$$

Pour $p \in \mathbb{C}$, on considère la suite $(u_n(p))_{n\in\mathbb{N}}$:

$$\left\{ \begin{array}{rcl} u_0(p) & = & p \\ u_{n+1}(p) & = & f(u_n(p), p) \end{array} \right.$$

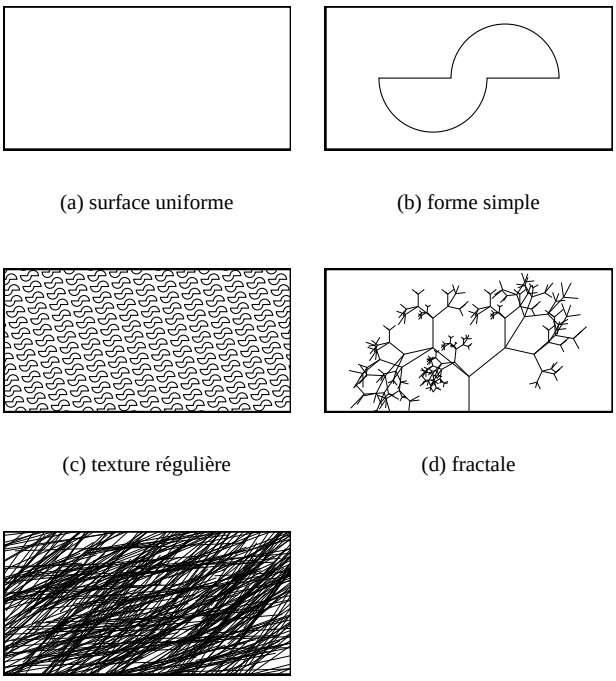
La fractale est alors l'ensemble :

$$E_{f,N} = \{p \in \mathbb{C} / \lim_n N(u_n(p)) \neq +\infty\}$$

C'est l'itération de la suite $(u(p)_n)_n$ qui fait naître la complexité de l'ensemble $E_{f,N}$. En particulier, si $f(.,p)$ est linéaire, alors l'expression de $u_n(p)$ est de complexité constante en n, donc $E_{f,N}$ n'est pas fractale.
Les fractales à temps d'échappement les plus simples qu'on puisse définir dépendent d'un paramètre $c \in \mathbb{C}$ tel que $|c| < 2$:

$$\left\{ \begin{array}{rcl} f(z,p) & = & z^2 + c \\ N(z) & = & |z| \end{array} \right.$$

C'est la *famille des fractales de Julia*. Par curiosité, on en



(a) surface uniforme (b) forme simple

(c) texture régulière (d) fractale

(e) forme aléatoire

FIG. 1: Structures géométriques de faible complexité donnant des figures de complexités perceptuelles croissantes.

FIG. 2: x varie de $1523 \times 2^{30} - 50$ à $1523 \times 2^{30} + 149$. La colonne correspondant à x porte sa décomposition binaire, $\text{blanc} = 0$, $\text{noir} = 1$, poids faibles vers le haut. La figure est fractale.

déduit la *fractale de Mandelbrot* :

$$\begin{cases} f(z,p) &= z^2 + p \\ N(z) &= |z| \end{cases}$$

2 Rendu des fractales à temps d'échappement

La méthode traditionnelle pour visualiser ces fractales (nommée BSM par Peitigen [3]) est simple :

– choisir un sous-ensemble borné de $\mathbb{C}$ (un rectangle), et définir une grille dessus (un quadrillage rectangulaire),
– pour chaque point p de la grille, calculer les valeurs de la suite jusqu'à ce que $N(u_n(p))$ dépasse une certaine valeur $d_{\max}$ ou qu'on ait effectué un certain nombre $n_{\max}$ d'itérations.

Cette méthode permet d'obtenir des images *bitmap* spectaculaires [2]. On peut colorer les points extérieurs de l'ensemble en fonction du rang m de la suite à partir duquel $N(u_m(p)) > d_{\max}$ (figure 3). [1]

On peut facilement rendre cet algorithme rendre parallèle : il n'y a pas de dépendance entre les pixels, et la quantité d'information à transporter très faible. Nous avons développé une architectrure client/serveur pour calculer des images sur 50 machines Sparc à l'ENSEEIHT. Le programme AltiVecFractal [11] exploite les instructions vectorielles présentes sur les processeurs G4.

Cependant, sur une machine monoprocesseur, avec une taille d'image assez conséquente, et un nombre d'itérations $n_{\max}$ de l'ordre de quelques milliers, on est loin de pouvoir faire un rendu temps réel.

C'est pour cette raison qu'ont été développées des stratégies d'accélération très efficaces. Elles visent surtout à ne pas perdre de temps sur les grandes zones untiformes. La présentation que nous en faisons ici est inspirée du programme Fractint [9].

2.1 Devinette

La méthode de *guessing* se décompose en deux passes :

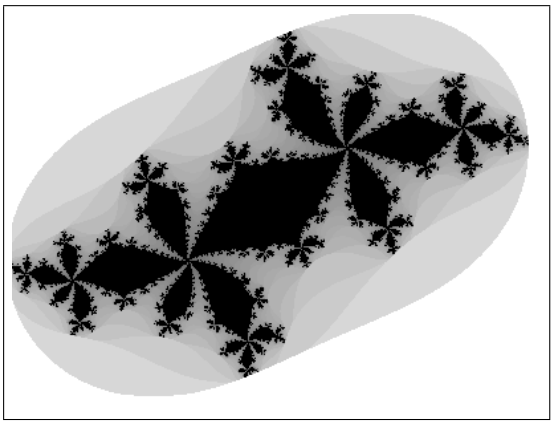
1. On calcule les valeurs de m sur une grille sous-échantillonnée.
2. Pour chaque point calculé, on regarde s'il a la même valeur que ses voisins. Si oui, on « colorie » le pavé qui l'entoure avec cette valeur, sinon on calcule aussi l'intérieur du pavé.

Le *guessing* génère facilement des images imprécises, mais il est très rapide. On l'utilise typiquement pour la prévisualisation. On peut aussi l'appliquer plusieurs fois à des échelles de plus en plus fines.

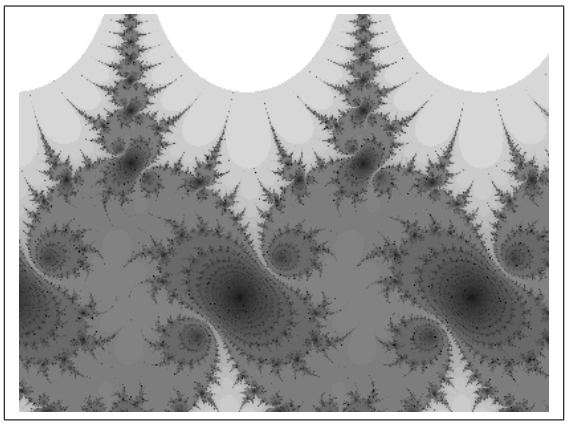
2.2 Pavage

Le pavage (*tesseral guess* ou méthode de Mariani) consiste à découper récursivement le rectangle auquel on s'intéresse en deux parties, et à calculer m sur les bords de chaque

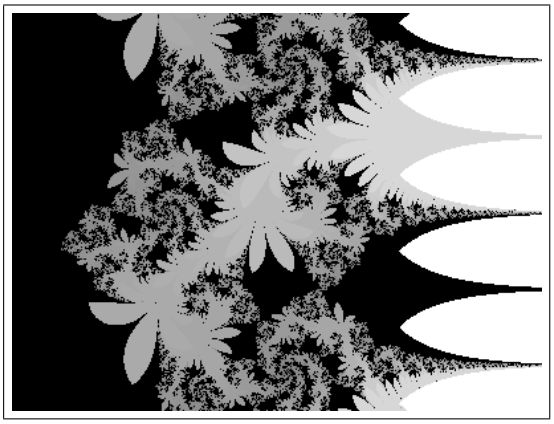
[1] Dans le cas des fractals de Mandelbrot et de Julia, on montre [4] que $\lim_n N(u_n(p)) = +\infty \iff \exists m \in \mathbb{N} / \, |u_m(p)| > 2$, ce qui donne tout de suite la valeur appropriée $d_{\max} = 2$.



(a) La fractale de Julia correspondant à $c = -0.515 - 0.567i$



(b) $f(z,p) = (1 + 0.1i)\sin z$, $N(z) = |\operatorname{Re} z|$. La fractale est périodique de période 2π, et sa surface est nulle.



(c) $f(z,p) = (0.9 + 0.3i)\exp z$, $N(z) = \min(|\operatorname{Re} z|, |\operatorname{Im} z|)$. La période est $2i\pi$.

FIG. 3: Exemples de fractales rendues avec la méthode classique. Les points extérieurs de la fractale sont plus sombres si m est plus grand. L'intérieur est en noir.

rectangle. Si les bords sont uniformes, on « colorie » tout l'intérieur du rectangle avec la valeur, sinon on redivise.
Cette méthode marche bien si on a des informations *ad hoc* qui nous empèchent de négliger les ensembles bornés isolés. Nous avons proposé une optimisation dans laquelle on choisirait judicieusement à quelle hauteur on fait la coupe, au lieu de le faire systématiquement au milieu.

2.3 Suivi de bords

La méthode de *boundary trace* (MBSM dans la terminologie de [3]) explore les bords de l'ensemble

$$F_m = \{p \in \mathbb{C} / u_m(p) < d_{\max}\}$$

pour un m donné. On génère une suite de points $(\pi_n)_n$ en marchant par des pas entiers dans les quatre directions canoniques, en s'assurant qu'on garde toujours l'intérieur de l'ensemble sur la gauche. La grille délimite des cases qui sont considérées comme faisant partie de F_m ou pas selon la situation de leur coin inférieur gauche (figure 4).

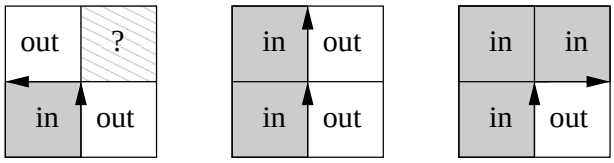


FIG. 4: À l'étape courante, la case à gauche est *in* et celle à droite *out*. Selon la situation des cases de devant, on tournera de manière à maintenir cette propriété.

On trouve ainsi à peu de frais les bords de la fractale, mais certaines régions connexes, dont le lien avec la région principale est trop fin, peuvent nous échapper.

2.4 Détection d'orbites

Après un régime transitoire plus ou moins long, les éléments de la suite $(u_n(p))_n$ se rapprochent souvent d'une spirale logarithmique (on a une vraie spirale logarithmique dans le cas linéraire). En estimant les paramètres de cette spirale, on peut déterminer si elle converge vers un attracteur ou si elle diverge. Si elle converge, on peut arrêter de calculer les éléments de la suite.
Cette méthode est très performante, mais elle échoue quelquefois pour des fonctions trop instables.

En somme, en combinant ces techniques programmées de manière astucieuse, on peut calculer les fractales en faisant 20 à 200 fois moins de calculs qu'avec l'implantation naïve. On peut ensuite zoomer et déplacer la fenêtre de visualisation en temps réel (voir XaOS [10]). On est cependant limité dans la profondeur du zoom par le *macheps*, qui fait qu'à un certain moment, les flottants sont trop peu précis pour représenter la différence entre deux pixels voisins.

3 Le rendu vectoriel

Regarder des fractales est intéressant, mais, par définition, on voit toujours des motifs répétitifs. C'est-à-dire que la partie originale du fractal est déjà perceptible à une échelle grossière et avec un faible nombre d'itérations.
C'est pour cette raison que nous avons voulu faire le rendu de F_m avec un nombre d'itérations m limité, mais en dessinant les bords de manière précise. Pour cela, il nous est apparu nécessaire d'avoir une représentation sous forme de polygones. Cette représentation vectorielle a pour avantage de respecter la structure de la fractale. Elle est éditable, et paraît parfaitement lisse, même quand elle est imprimée.

4 Suivi de bords vectoriel

En premier lieu, nous avons cherché à suivre les bords de l'ensemble F_m. Pour cela, il faut qu'une méthode *ad hoc* fournisse un point dans chaque région connexe de F_m et de $\mathbb{C}\backslash F_m$.
Pour trouver un point de départ du suivi, il suffit de chercher un point $p_0 \in \partial F_m$ par dichotomie entre $p \in F_m$ et $q \notin F_m$ (figure 5).

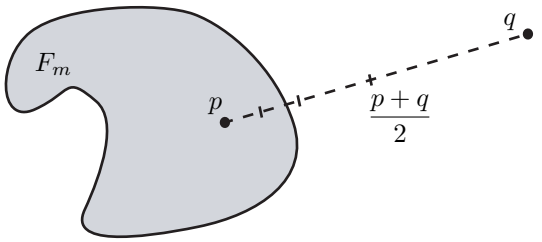


FIG. 5: Recherche du bord par dichotomie

p_0 est le premier terme d'une suite de points $(p_n)_n$ du bord. Nous avons développé deux méthodes pour calculer les termes suivants de la suite.

4.1 Méthode intuitive

Elle consiste à effectuer des pas, de longueur l. Si on a les points p_{n-1} et p_n, on cherche le point p_{n+1} sur l'arc de cercle de rayon l et d'angles $[-\alpha_{\max}, \alpha_{\max}]$ autour de la direction $\overrightarrow{p_{n-1}p_n}$ (figure 6).

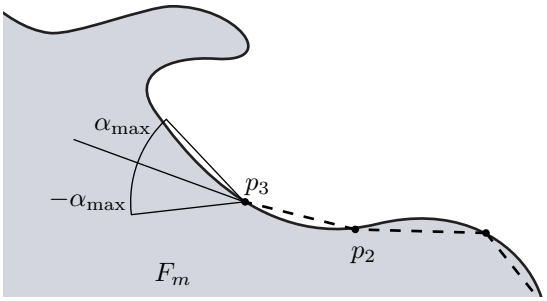


FIG. 6: Suivi du bord par pas de longueur constante

Si les deux extrémités de l'arc sont toutes deux à l'intérieur ou à l'extérieur, on double $\alpha_{\max}$ jusqu'à ce que ce ne soit plus le cas. On trouve ensuite l'intersection (ou une des intersections) de l'arc avec l'ensemble par dichotomie, ce qui nous donne p_{n+1}.
Cette méthode repose sur le choix d'un l pertinent en fonction du rayon de courbure de l'ensemble. Par exemple, on

voit sur la figure 6 que le pas est trop grossier pour suivre la protubérence du haut. Malheureusement, pour les fractales, le rayon de courbure varie très soudainement. Ceci introduit non seulement des erreurs de suivi, mais en plus, l'algorithme peut être tellement désorienté qu'il repasse sur un bord, ce qui le fait rentrer dans une boucle sans fin.
On pourrait adopter une méthode adaptative qui diminue l quand on détecte un boucalge. Cependant, cela aurait pour conséquence que les pas ne seraient pas de la même longueur, pour une courbure donnée, de part et d'autre d'une protubérence.

4.2 Méthode réaliste

Une méthode plus robuste s'appuie sur la technique présentée au 2.3, qui fournit déjà une suite de points π_n qui suit la courbe. Ces points ne sont pas *sur* la courbe. Par contre, chaque paire (π_n, π_{n+1}) correspond à une paire de points (π'_n, π''_n) de part et d'autre de la courbe : ce sont les coins inférieurs gauches des cases adjacentes. Il suffit donc de procéder par dichotomie pour trouver la suite des p_n (figure 7). Si la longueur du pas choisi entre les π_n est λ, la distance entre les p_n est inférieure à $\lambda\sqrt{2}$.

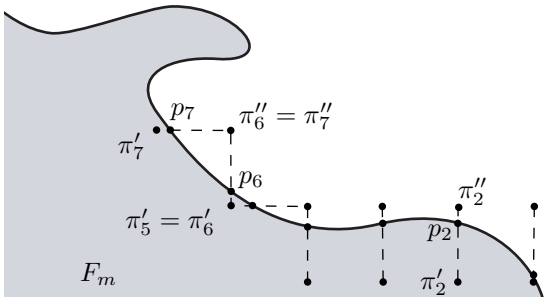


FIG. 7: Les points π'_n sont sur F_m, et les π''_n à l'extérieur. Chaque paire sert de support au calcul d'un point p_n.

Par construction de la suite $(\pi_n)_n$, cette méthode ne boucle pas. Malheureusement, elle ne découvre pas les détails de taille inférieure à λ. C'est cette raison qui nous a fait abandonner les méthodes de suivi génériques[2].

5 Itération inverse

L'itération inverse consiste partir de l'ensemble $\{p \in \mathbb{C}/N(p) < d_{\max}\}$, et d'en déduire F_m.

5.1 Cas où $f(z,p)$ ne dépend pas de p

Dans ce cas, on peut noter $f(z,p) = f(z)$, et on a :

$$\begin{aligned} u_m(p) &= f^m(p) \\ F_m &= \{p \in \mathbb{C}/N(u_m(p)) < d_{\max}\} \\ &= (N \circ u_m)^{-1}([0, d_{\max}[) \\ &= (f^m)^{-1}(N^{-1}([0, d_{\max}[)) \\ &= (f^{-1})^m(F_0) \end{aligned}$$

f^{-1} est l'inverse de f au sens ensembliste. $f^{-1}(z) = f^{-1}(\{z\})$ peut être vide, ou composé de plusieurs éléments.

[2]Génériques parce qu'elles peuvent s'appliquer (avec plus de succès) à des fonctions non-fractales

Pour calculer F_m, il suffit donc de partir de F_0 et de passer de F_n à F_{n+1} à l'aide de f^{-1}. En pratique, il faut faire deux choix :

1. quels points de $p_{0,1}, ..., p_{0,h} \in F_0$ va-t-on choisir initialement ?
2. à l'étape n, va-t-on calculer tous les $f^{-1}(p_{n,i})$, et sinon, lesquels gardera-t-on ?

Les algorithmes IIM-A et MIIM proposés dans [3] choisissent au hasard les $p_{0,i}$. Ensuite, ils choisissent au hasard (IIM-A) ou selon une étude statistique (MIIM) un des éléments de $f^{-1}(p_{n,i})$. On obtient finalement un nuage de points sans surface (figure 8).

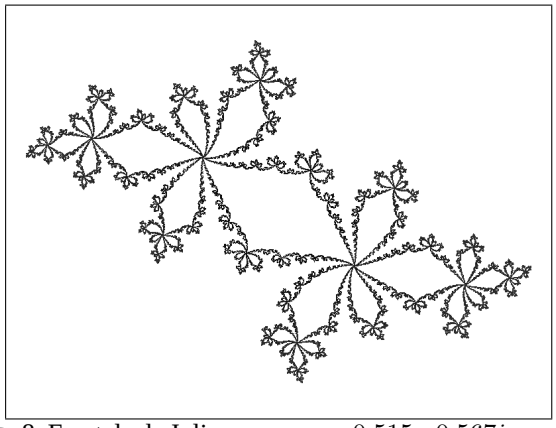


FIG. 8: Fractale de Julia, pour $c = -0.515 - 0.567i$, rendue avec la méthode MIIM.

La fractale de Julia

Dans notre approche, nous choisisions les $p_{0,i}$ sur ∂F_0, et nous suivons *tous* les $f^{-1}(p_{n,i})$.
Dans ce cas, $F_0 = \{p \in \mathbb{C}/ \mid p \mid < 2\} = \mathcal{D}(0,2)$ est le disque (ouvert) de centre 0 et de rayon 2. $f^{-1}(p) = \{\sqrt{p-c}, -\sqrt{p-c}\}$, en notant $\sqrt{z} = z^{1/2}$, où on choisit l'argument dans $[-\pi, \pi[$. Donc $f^{-1}(p)$ a deux éléments, sauf pour $f^{-1}(0) = 0$.
Nous utilisons un polygone régulier à h côtés pour représenter $\partial F0$, et à chaque étape nous calculons la transformée des sommets. Ceci signifie que nous devons stocker $h2^n$ points à l'étape n. La puissance des ordinateurs actuels nous limite à un nombre d'itérations de l'ordre de $m = 20$. Il faut aussi savoir comment relier les points les uns aux autres pour « coller » le mieux aux ∂F_n.

Topologie de F_n

Les F_n sont emboîtés : $F_0 \supset F_1 \supset \ldots \supset F_m$.

Démonstration : Montrons que $F_1 \subset F_0$. Soit $q \in F_1$. Alors $p = f(q) \in F_0$, donc $|p| < 2$.

$$\begin{aligned} |p-c| &< |p| + |c| < 2 + 2 = 4 \\ \sqrt{|p-c|} &< 2 \\ |\sqrt{p-c}| &< 2 \quad \text{(passage aux complexes)} \\ |q| \in |f^{-1}(p)| &\subset \mathcal{C}(0,2) \end{aligned}$$

Donc $q \in F_0$, donc $F_1 \subset F_0$. Par récurrence, $F_n \subset F_{n+1}$ ■

L'image inverse d'un ensemble connexe ouvert K par la fonction $\kappa(z) = z^2$ est soit un ensembe connexe ouvert, soit deux ensembles connexes ouverts.

Démonstration : soient $a_1, a_2 \in K$. K est connexe, donc il y a un chemin[3] γ de a_1 vers a_2. γ coupe le demi-axe des réels négatifs en n points (n est éventuellement nul, l'ouverture de K assure qu'on peut trouver γ tel que n ne soit pas infini) :

$$0 = t_0 < t_1 < t_2 < \ldots < t_n < t_{n+1} = 1$$

$$\forall k = 1..n,\ \ \gamma(t_k) \in \mathbb{R}^{-*}$$

On définit γ_+ sur chaque tronçon $[t_k, t_{k+1}[$. Pour $t \in [t_k, t_{k+1}[$,

$$\gamma_+(t) = (-1)^k \sqrt{\gamma(t)}$$

$\sqrt{.}$ est continu et infiniment dérivable partout, sauf sur la demi-droite des réels négatifs, donc γ_+ est continu et continûment dérivables par morceaux sur $[0,1]\backslash\{t_1, \ldots, t_n\}$.

Montrons que γ_+ est continu en t_k. Les continuités à gauche et à droite donnent :

$$\lim_{t \to t_k^-} \gamma_+(t) = (-1)^k \sqrt{\gamma(t_k)}$$

$$\lim_{t \to t_k^+} \gamma_+(t) = -(-1)^{k+1} \sqrt{\gamma(t_k)}$$

Les deux limites sont les mêmes, donc γ_+ est continu en t_k. Donc γ_+ est un chemin de $\sqrt{a_1}$ vers $(-1)^n\sqrt{a_2}$. De la même manière, $\gamma_- = -\gamma_+$ est un chemin de $-\sqrt{a_1}$ vers $(-1)^{n+1}\sqrt{a_2}$

Le résultat à retenir est : l'image inverse d'un chemin de K par κ est deux chemins : $\kappa^{-1}(\gamma(t)) = \{\gamma_+(t), \gamma_-(t)\}$.

▷Montrons que $\kappa^{-1}(K)$ comprend au plus deux composantes connexes. Soient $b_1, b_2, b_3 \in \kappa^{-1}(K)$, il faut montrer que deux des trois sont sur un ensemble connexe, donc qu'il existe un chemin les reliant.

Notons γ_+ et γ_- (resp. δ_+ et δ_-) les composants de l'inverse d'un chemin de κb_1 vers $\kappa(b_2)$ (resp. $\kappa(b_2)$ vers $\kappa(b_3)$) tels que $\gamma_+(0) = b_1$ (resp. $\delta_+(0) = b_2$). Si $\gamma_+(1) = b_2$, γ_+ est un chemin reliant b_1 à b_2.

Sinon $\gamma_+(1) = \delta_-(0)$. Si $\delta_-(1) = b_3$ alors le chemin constitué de γ_+ et δ_- mis bout à bout relie b_1 à b_3. Sinon, $\delta_+(1) = b_3$, donc le chemin δ_+ relie b_2 à b_3. ■

Si il existe un lacet de K qui « entoure » 0 (c'est-à-dire que 0 est d'indice non nul par rapport au lacet), alors il n'y a qu'une composante connexe.

Démonstration : Appelons γ le lacet, $\gamma(0) = \gamma(1) = a$, et γ_+ et γ_- ses antécédants.

Soient $b_1, b_2 \in \kappa^{-1}(K)$. Il y a un chemin δ de $\kappa(b_1)$ vers a, et ϵ de a vers $\kappa(b_2)$. Les images inverses sont δ_+, δ_- et ϵ_+, ϵ_-, tels que $\delta_+(0) = b_1$ et $\epsilon_+(1) = b_2$. Alors si $\delta_+(1) = \epsilon_+(0)$, le chemin constitué de δ_+ et ϵ_+ mis bout à bout relie b_1 à b_2.

Sinon, $\delta_+(1) = -\epsilon_+(0) = \pm\sqrt{a}$. Si $\delta_+(1) = \sqrt{a}$ (resp. $= -\sqrt{a}$) alors le chemin constitué de δ_+, γ_+ (resp. γ_-) et ϵ_+ mis bout à bout relie b_1 à b_2.

Donc $\kappa^{-1}(K)$ est connexe.

▷Inversément, supposons que $\kappa^{-1}(K)$ est connexe. κ^{-1} est aussi symétrique par rapport à 0. .... ■

$f^{-1}(z) = \kappa^{-1}(z - c)$, et les F_n sont des surfaces simplement connexes, donc F_n est connexe si et seulement si $c \in F_n$. Sinon, F_{n+1} est constitué de deux ensembles connexes symétriques par rapport à l'origine.

En combinant ceci avec la propriété d'emboîtement, on voit que les F_n sont connexes jusqu'à un certain rang n_{conn} (éventuellement infini), premier rang tel que $c \notin F_{n_{\text{conn}}}$, au-delà duquel ils se divisent perpétuellement en deux. F_n est donc composé de

$$\begin{cases} 1 & \text{si } n < n_{\text{conn}} \\ 2^{n-n_{\text{conn}}} & \text{sinon} \end{cases}$$

composantes connexes.

L'algorithme

Pour tracer les polygones, on procède donc en remplissant un tableau $p(n, j)$ où n est le numéro de l'itération, et $j = 1..h2^n$ est un numéro de point.

Pour j **dans** 0 **à** $h - 1$ **faire**
  $p(0, j) := 2e^{\frac{2i\pi j}{h}}$
Fin pour
$n_{\text{conn}} := m$
Pour n **dans** 1 **à** m **faire**
  $n_{\text{coupes}} := 0$
  Pour j **dans** 0 **à** $h2^{n-1} - 1$ **faire**
    Si $[p(n-1, j-1)\ p(n-1, j)]$ coupe $\mathbb{R}^-$ **alors**
      $n_{\text{coupes}} := n_{\text{coupes}} + 1$
    Fin si
    $z := \sqrt{p(n-1, j) - c}$
    Si n_{coupes} pair **alors**
      $p(n, j) := z$ et $p(n, 2^{n-1} + j) := -z$
    Sinon
      $p(n, j) := -z$ et $p(n, 2^{n-1} + j) := z$
    Fin si
  Fin pour
  Si (n_{coupes} pair) $\wedge (n_{\text{conn}} = m)$ **alors**
    $n_{\text{conn}} := n$
  Fin si
Fin pour

Pour tracer les polygones, on procède ainsi :
- Si $n \leq n_{\text{conn}}$, les points de $p(n, .)$ forment un polygone.
- Sinon, les points de $(p(n, .)$, regroupés par paquets de $h2^{n_{\text{conn}}}$ forment des polygones disjoints.

Expérimentations

Nous avons développé en C un programme qui permet de visualiser rapidement les fractales tracées ainsi (figure 9). On peut également sauvegarder les fichiers au format EPS, qui est facilement utilisable dans des logiciels de dessin vectoriel. La figure 10 présente quelques exemples.

[3] Un *chemin* est une fonction $\gamma : [0,1] \mapsto K$ continue continûment dérivable par morceaux. Si $\gamma(0) = \gamma(1)$, c'est un *lacet*. Les termes employés ici sont issus de [5].

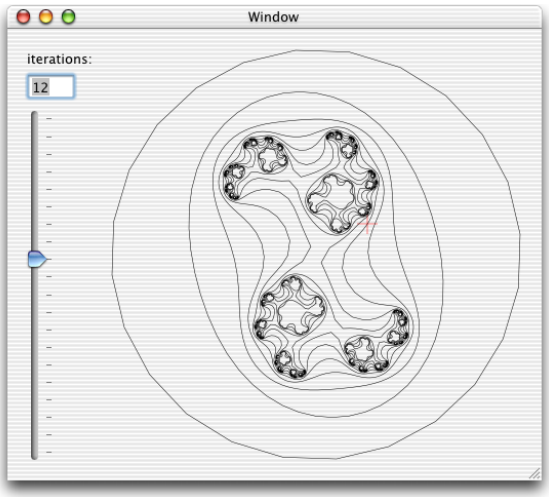
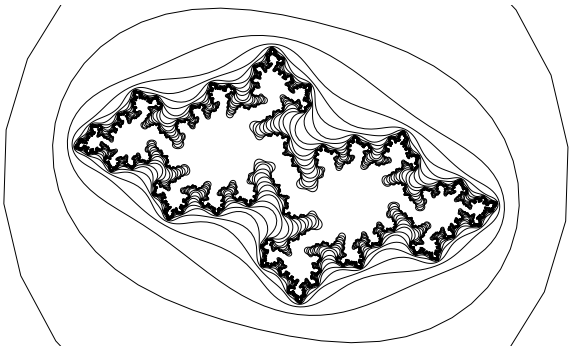
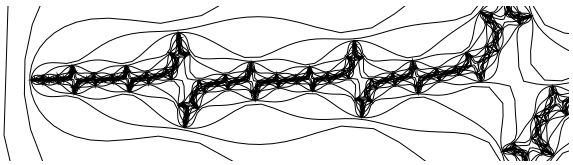


FIG. 9: Capture d'écran de JULIARING. La croix rouge représente le point c.



(a) pour $c = -0.69 + 0.38i$, et $m = 20$



(b) pour $c = -1.66 + 0.04i$, et $m = 15$ (détail)

FIG. 10: Exemples de fractales de Julia.

5.2 Pour l'ensemble de Mandelbrot

Pour l'ensemble de Mandelbrot, il n'y a pas de propriété facilement exploitable comme pour les fractales de Julia.

Comme précédemment, nous partons de $F_0 = \mathcal{D}(0, 2)$. Les images des itérées F_n sont connexes et emboîtées. Le premier résultat a été démontré par Douady et Hubbard [6], le second est analogue au cas des fractales de Julia. Dans ce contexte, il n'y a pas de problèmes pour connecter les points.

Il faut donc trouver $F_m = u_m^{-1}(F_0)$, soit en pratique $\{p_{m,j}, p_{m,j+h}, ..., p_{m,j+2^m-1}\} = u_m^{-1}(p_{0,j})$. Il n'y a pas de raison pour calculer tous les F_n.

Les polynômes

$u_m(p)$ est un polynôme de degré 2^m en p : $u_m \in \mathbb{N}_{2^m}[X]$. Les coefficients de ce polynôme n'ont rien de particulier (figure 11).

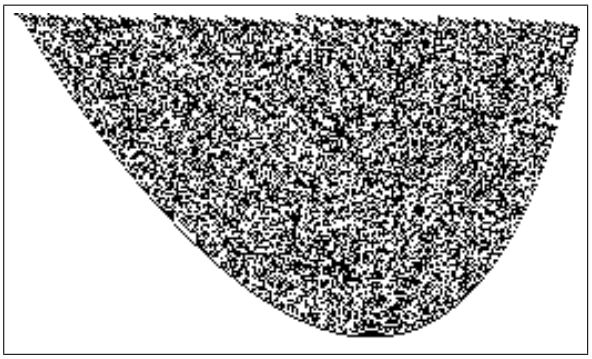


FIG. 11: Décomposition binaire des coefficients du polynôme u_8. On ne voit rien de régulier qui pourrait être exploité.

Trouver les racines d'un polynôme de degré 2^m est une opération en $\mathcal{O}(m2^m)$ au moins, et il faut la réitérer pour tous les $h2^m$ points, ce qui est trop coûteux, même pour un petit m.

Méthode itérative

Nous exploitons le fait que pour p donné, on peut obtenir la dérivée $u'_m(p)$ en plus de $u_m(p)$. D'après les lois dérivation des fonctions analytiques,

$$\begin{cases} u'_0(p) &= 1 \\ u'_{n+1}(p) &= f'(u_n(p), p)u'_n(p) = 2u_n(p)u'_n(p) \end{cases}$$

en notant f' la dérivée de f par rapport à son premier argument.
On peut utiliser u'_m pour appliquer la méthode itérative de Newton selon l'algorithme :

Pour j **dans** 0 **a** $h-1$ **faire**
  $p(0, j) := 2e^{\frac{2i\pi j}{h}}$
Fin pour
Pour j **dans** 0 **a** $h2^{n-1} - 1$ **faire**
  $p := p(m, j-1)$
  Faire
    $p := p - (p(0, j \mod h) - u_m(p))/u'_m(p)$
    $e := |p(0, j \mod h) - u_m(p)|$
  Tant que $e > \varepsilon$
  $p(m, j) := p$
Fin pour

ε est une erreur admissible pour e.

Expérimentations

Nous avons implanté l'algorithme en C étendu (gcc gérant les complexes). Nous avons pris $\varepsilon = 10^{-18}$, un peu au dessus du *macheps* pour les flottants 64-bits. L'algorithme converge sans problèmes pour $m < 7$. Ensuite, on est obligé d'augmenter le nombre de côtés h du polygone selon une loi empirique $h = 100 \times 2^{m-6}$. La figure 12 représente le résultat.

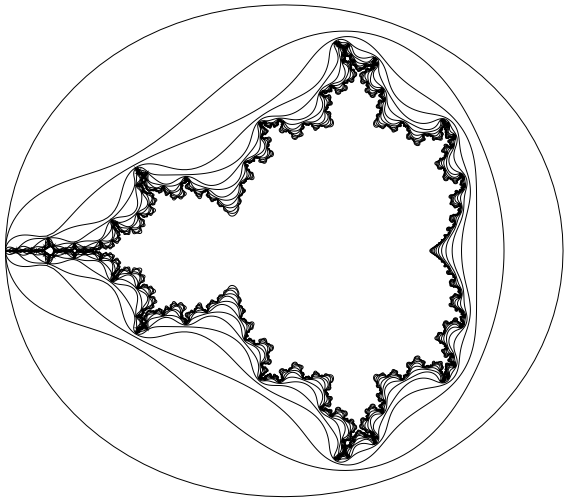


FIG. 12: Approximation des bords de F_m pour $m = 1..13$.

Conclusion

Les fractales sont bien connues maintenant, et très intensivement étudiées sur le plan théorique. Cependant, les méthodes de visualisation restent en général assez primitives. Notre contribution consiste donc principalement en une méthode pour dessiner les bords, et potentiellement l'intérieur de fractales à temps d'échappement sous forme de polygones.

Dans la suite de notre travail, plusieurs voies s'offrent à nous :

– Calculer des parties de polygones, c'est-à-dire pouvoir zoomer sur une région sans calculer la partie du polygone invisible.
– Gérer des bords en splines, au lieu de polygones dont les angles sont trop apparents ou, à défaut, adapter la longueur des segments.
– Approcher aussi la frontière de la fractale de Mandelbrot par l'intérieur.
– Peut-être une généralisation à la 3D ?

Ce dernier point fait référence aux recherches en cours sur les fractales en 3 et 4-D (le premier étant une coupe du second). Elles se basent sur des généralisations des complexes, comme les quaternions et les hypercomplexes exploités dans POV [8] ou les bicomplexes [7].

D'une manière générale, les logiciels de dessin vectoriel ignorent complètement les fractales, alors que celles-ci ont un intérêt esthétique indéniable.

Références

[1] B. Mandelbrot, *Les objets fractals*, Flammarion, Paris, 1975.

[2] H-O. Peitigen, P.H. Richter *The Beauty of Fractals*, Springer-Verlag, Berlin, 1986.

[3] H.-O. Peitigen, D. Saupe *The Science of Fractal Images*, Springer-Verlag, Berlin, 1988.

[4] R. Devaney, *Chaos, Fractals, & Dynamica, Computer-experimenten in de wiskunde*, Addison-Wesley, Amsterdam, 1992.

[5] *Dictionnaire des mathématiques*, article sur les fonctions analytiques, J.-L. Verley, Encyclopædia Universalis/Albin Michel, Paris, 1997.

[6] Douady, A., Hubbard, J. H. *Itération des polynômes quadratiques complexes*, CRAS Paris, 1982.

[7] D. Rochon, *A Generalized Mandelbrot Set for Bicomplex Numbers*, World Scientific/Fractals, Volume 8, Number 4, december 2000.

[8] *The Persistence of Vision Raytracer*, (`http://www.povray.org`).

[9] Timothy Wegner, Jonathan Osuch, George Martin, Robin Bussel et al., FRACTINT, (`http://www.fractint.org`).

[10] Jan Hubicka et al., XAOS (`http://www.gnu.org/software/xaos`).

[11] Dauger Research, ALTIVECFRACTAL (`http://daugerresearch.com`).