Méthodes itératives

Exemples de courbes fractales

Arguments mathématiques Entre deux compacts non vides du plan (ou plus généralement, d'un evn de dimension finie), on peut définir la distance de Hausdorff : $d(K1,K2)=\max_{x \text{ dans } K1}(d(x,K2))$.

L'ensemble **H** des compacts non vides du plan n'est pas un espace vectoriel normé (ce n'est pas un espace vectoriel !) mais cette distance permet de définir la notion de limite, de suite convergente, et de suite de Cauchy. On peut montrer que les suites de Cauchy de **H** convergent : on dit que **H** est complet.

Conséquence importante : les fonctions k-lipschitziennes avec k<1 (« contractantes »), admettent nécessairement un « point » fixe unique, et toute suite récurrente $K_{n+1} = f(K_n)$ définie avec une application contractante converge dans H, vers un compact qui ne dépend pas des conditions initiales.

Or, il est facile de trouver de telles f : par exemple, les transformations affines associées à une application linéaire F telle que |det F|<1 conviennent.

Données de l'algorithme

Une figure initiale donnée par une liste de points (un point est une liste de deux coordonnées).
Une liste de transformations ; une transformation affine est donnée par une liste de 6 coefficients tels que :

```
x'= a[1]x+a[2]y+a[3], y'=a[4]x+a[5]y+a[6]
```

Principe de l'algorithme

On applique la liste de p transformations à la figure initiale (segment, triangle, quadrilatère, etc.). On obtient une liste de p figures. On réitère le processus et on obtient une liste de p^n figures après n itérations.

Implémentation MAPLE

Le tracé nécessite le package « plots » :

```
> with(plots):
```

Procédure calculant l'image d'un point par une transformation :

```
> appliquerpoint:=proc(a,M)
[a[1]*M[1]+a[2]*M[2]+a[3],a[4]*M[1]+a[5]*M[2]+a[6]]
end;
```

Procédure calculant l'image d'une figure (liste de points) par une transformation :

```
> appliquer:=proc(a,L::list)
local i;
[seq(appliquerpoint(a,L[i]),i=1..nops(L))]
end proc ;
```

Figure initiale (donnée par une liste de points) :

```
> figure0:=[[0.,0.],[1.,0.]];
```

Procédure principale :

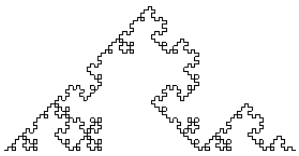
```
> fractal:=proc(n,transformations ::list)
local i,j,k,figures;
global figure0; #figure initiale
     figures:=figure0; # initialisation
     # images itérées de la figure initiale
     for j to n do figures :=
        seq(seq(appliquer(transformations[i],[figures][k]),
        k=1..nops([figures])), i=1..nops(transformations));
     end do;
     # calcul de la figure obtenue au rang n
     polygonplot([figures],axes=none,color=black,scaling=
     constrained)
end proc;
```

Exemple

Déterminer 5 transformations affines simples qui transforment le segment unité en la figure suivante (chaque segment est de longueur 1/3) :

Solution :

```
> v:=evalf(1/3) :fractal(4,[[v,0,0,0,v,0],
[0,-v,v,v,0,0],[v,0,v,0,v,v],[0,-v,2*v,v,0,0],[v,0,2*v,0,v,0]]);
```



Exemple : tamis de Sierpinski

On applique des transformations au triangle isocèle rectangle suivant:

```
> figure0:=[[0,0.],[1.,0.],[0,1.]];
> fractal(6,[[.5, 0, 0, 0, .5, 0], [.5, 0,.5, 0, .5,  0], [.5,
0,.25, 0, .5,  .5]]);
```