Tronc Commun Maple
Tracé de fractales élémentaires

Corrigé

16 mars 2006

1 Tracé de courbes fractales élémentaires

1.1 Fractales Géométriques

1.1.1 Le flocon de von Koch

1.
```
koch2 := proc(p : :numeric,h : :numeric,a : :numeric,o)
local x,y,s,pts,n,t ;
    s :=a ;
    # on construit une table contenant les points du flocon
    pts :=table([]) :
    # on se place à l'origine
    x :=o[1] ; y :=o[2] ;
    n :=0 ;
    if p=1 then
        # ranger les points du générateur d'ordre 1
        # et d'angle initial s dans la table
        for t in [1,-2,1] do
            pts[n] := [x,y] ;
            x := evalhf(x+cos(Pi/3*s)*h) ;
            y := evalhf(y+sin(Pi/3*s)*h) ;
            s :=s+t ;
            n :=(n+1) ;
        od ;
        pts[n] := [x,y] ;
        # renvoyer le contenu de la table représentant le générateur d'ordre 1
        [seq(pts[i],i=0 .. n)] ;
    else
        # se placer aux positions des points du générateur de taille 1, et
        # tracer à cet endroit un générateur d'ordre p-1
        for t in [1,-2,1] do
            pts[n] := koch2(p-1,h/3,s,[x,y]) ;
            x := evalhf(x+cos(Pi/3*s)*h) ;
            y := evalhf(y+sin(Pi/3*s)*h) ;
            s :=s+t ;
            n :=(n+1)
        od ;
```

```
            pts[n] := koch2(p-1,h/3,s,[x,y]) ;
            # renvoyer le contenu de la table représentant le générateur
            # de taille h.
            [seq(op(pts[i]),i=0..n)] ;
        fi ;
    end ;
```

2. On remarque que la taille d'un segment de la courbe est 3^{-p}, où p est l'ordre de la courbe.

```
koch :=proc(n)
    plot(koch2(n,1,0,[0,0]), scaling=constrained, style=LINE, axes=NONE) ;
end ;
```

1.1.2 L'ensemble de Mandelbrot

–

```
couleur :=proc(a,b)
local x,y,xi,yi,n ;
    x :=a ;
    y :=b ;
    for n from 0 to 30 while evalf(x^2+y^2) < 2 do ;
        xi :=evalf(x^2-y^2+a) ;
        yi :=evalf(2*x*y+b) ;
        x :=xi ;
        y :=yi ;
    od ;
    n ;
end ;
```

–

```
plot3d(0,(-2)..(1),(-1)..(1),orientation=[-90,0],
       style=patchnogrid, scaling=constrained,axes=framed,
       numpoints=20000,color=couleur) ;
```

1.1.3 Les ensembles de Julia

–

```
couleur :=proc(a,b)
local x,y,xi,yi,n ;
global reel,imaginaire ;
    x :=a ;
    y :=b ;
    for n from 0 to 100 while evalf(x^2+y^2)<3 do ;
        xi :=evalf(x^2-y^2+reel) ;
        yi :=evalf(2*x*y+imaginaire) ;
        x :=xi ;
        y :=yi ;
    od ;
    n ;
end :
```

–

```
reel :=-1.25 ;
imaginaire :=0.0 ;

plot3d(0,(-13/10)..(13/10),(-13/10)..(13/10),orientation=[-90,0],
       style=patchnogrid,scaling=constrained,axes=framed,
       numpoints=20000,color=couleur) ;
```

2 Plus longue sous-suite commune (PLSC)

1. Pour faire une recherche naïve, on va tester toutes les sous-suites de $(x_k)_{k=0}^n$ qui sont au nombre de 2^n...
2. On a la propriété de sous-structure suivante : si $(u_k)_{k=0}^p$ est une PLSC de $(x_k)_{k=0}^n$ et $(y_k)_{k=0}^m$, alors :
   - Si $x_n = y_m$, alors $u_p = x_n = y_m$ (sinon on peux construire une sous-suite commune plus longue), et $(u_k)_{k=0}^{p-1}$ est une PLSC de $(x_k)_{k=0}^{n-1}$ et $(y_k)_{k=0}^{m-1}$: c'est une sous-suite commune, et si on en trouve une trouve une plus longue on peux construire une sous-suite commune de $(x_k)_{k=0}^n$ et $(y_k)_{k=0}^m$ plus longue que $(u_k)_k$.
   - Si $x_n \neq y_m$ et $u_p \neq x_n$, alors $(u_k)_{k=0}^p$ est une PLSC de $(x_k)_{k=0}^{n-1}$ et $(y_k)_{k=0}^m$
   - Si $x_n \neq y_m$ et $u_p \neq y_m$, alors $(u_k)_{k=0}^p$ est une PLSC de $(x_k)_{k=0}^n$ et $(y_k)_{k=0}^{m-1}$
3. On a donc la formule de récurrence suivante, en notant l[i,j] la longueur d'une PLSC de $(x_k)_{k=0}^i$ et $(y_k)_{k=0}^j$:

$$l[i,j] = \begin{cases} 0 & \text{si } i = 0 \text{ ou } j = 0 \\ l[i-1, j-1] + 1 & \text{si } i > 0, j > 0 \text{ et } x_i = y_j \\ max(l[i, j-1], l[i-1, j]) & \text{si } i > 0, j > 0 \text{ et } x_i \neq y_j \end{cases}$$

```
longueur_PLSC := proc(x,y)
local n,m,i,j,T;
n := nops(x);
m := nops(y);
for i from 0 to n do
  T[i,0] := 0;
od;
for j from 0 to m do
  T[0,j] := 0;
od;
for i from 1 to n do
  for j from 1 to m do
    if x[i] = y[j] then
      T[i,j] := T[i-1,j-1]+1;
    else
      T[i,j] := max(T[i-1,j], T[i,j-1])
    fi;
  od;
od;
RETURN (T[n,m]);
end :
```

4.

```
PLSC := proc(x,y)
local n,m,i,j,T;
n := nops(x);
m := nops(y);
for i from 0 to n do
  T[i,0] := [];
od;
for j from 0 to m do
  T[0,j] := [];
od;
for i from 1 to n do
  for j from 1 to m do
    if x[i] = y[j] then
      T[i,j] := [op(T[i-1,j-1]),x[i]];
    else
      if nops(T[i-1,j]) > nops(T[i,j-1]) then
        T[i,j] := T[i-1,j];
      else
```

```
        T[i,j] := T[i,j-1] ;
      fi ;
    fi ;
  od ;
od ;
RETURN (T[n,m]) ;
end :
```

5. À partir du tableau des longueurs, on peux retrouver dans quel cas de la propriété de sous-structure on se trouve en regardant les longueurs de certaines PLSC, et ainsi trouver le dernier élément de la PLSC :

```
PLSC_2 := proc(x,y)
local n,m,i,j,T,u ;
n := nops(x) ;
m := nops(y) ;
for i from 0 to n do
  T[i,0] := 0 ;
od ;
for j from 0 to m do
  T[0,j] := 0 ;
od ;
for i from 1 to n do
  for j from 1 to m do
    if x[i] = y[j] then
      T[i,j] := T[i-1,j-1]+1 ;
    else
      T[i,j] := max(T[i-1,j], T[i,j-1])
    fi ;
  od ;
od ;
u :=[] ;
i :=n ; j :=m ;
while (i <> 0 and j<>0) do
  if x[i] = y[j] then
    u := [x[i],op(u)] ;
    i := i-1 ;
    j := j-1 ;
  else
    if T[i-1,j] > T[i,j-1] then
      i := i-1 ;
    else
      j := j-1 ;
    fi ;
  fi ;
od ;
RETURN(u) ;
end :
```

L'algorithme obtenu demande un espace mémoire mn, et un temps calcul $O(mn)$.