Dessiner des fractales

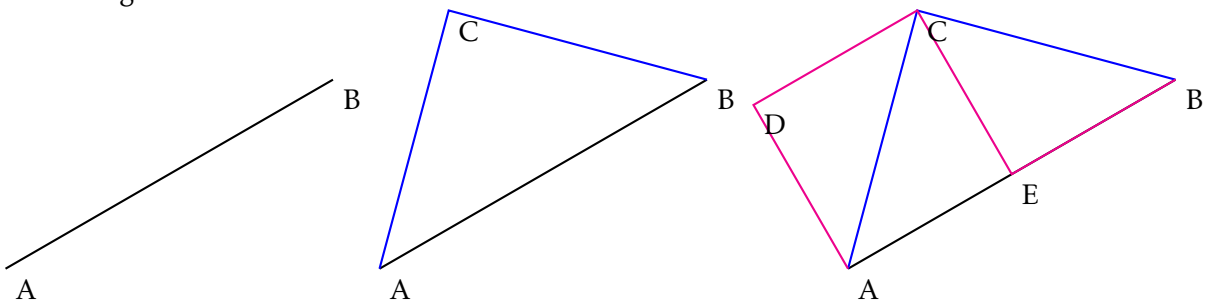
Objectif : L'objectif de ce tp est :

- d'utiliser des algorithmes récursifs pour tracer des fractales
- de déterminer les stratégies à adopter pour afficher les bons segments

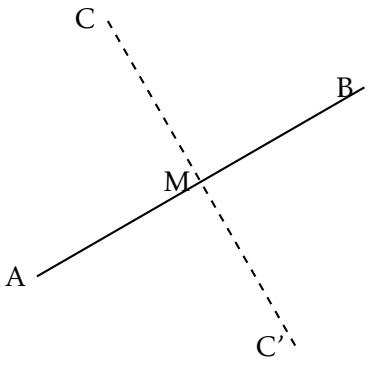
1 Courbes du dragon

1.1 Motif

Pour tracer la courbe du dragon, chaque segment est décomposé en deux segments permettant de former un triangle isocèle rectangle où l'hypoténuse est le segment au rang n et les deux côtés sont les segments au rang $n+1$. En suivant les segments dans l'ordre, il y a alternance du côté d'apparition des nouveau segments à chaque changement de segment.



1.2 Point de vue géométrique



$$\overrightarrow{AC} = \overrightarrow{AM} + \overrightarrow{MC} \quad \text{avec} \quad \overrightarrow{AM} = \frac{1}{2}.\overrightarrow{AB} \quad ; \quad \left\|\overrightarrow{MC}\right\| = \left\|\overrightarrow{MB}\right\| \quad \text{et} \quad \overrightarrow{MC}.\overrightarrow{MB} = 0$$

$$\overrightarrow{AM} = \left(\frac{x_B - x_A}{2}; \frac{y_B - y_A}{2}\right) \quad \text{et} \quad \overrightarrow{MC} = \left(-\frac{y_B - y_A}{2}; \frac{x_B - x_A}{2}\right)$$

$$\begin{aligned} C &= \left(x_A + \frac{x_B - x_A}{2} - \frac{y_B - y_A}{2}; y_A + \frac{y_B - y_A}{2} + \frac{x_B - x_A}{2}\right) \\ &= \left(\frac{x_B + x_A}{2} - \frac{y_B - y_A}{2}; \frac{y_B + y_A}{2} + \frac{x_B - x_A}{2}\right) \end{aligned}$$

Q - 1 : *Écrire une fonction* `nouv_pts_dragon(A, B, sign)` *qui prend en argument deux tuples* `A` *et* `B` *contenant les coordonnées cartésiennes des points* `A` *et* `B` *ainsi qu'un troisième argument* `sign` *valant soit* `1` *soit* `-1` *et qui renvoie dans un tuple les coordonnées du point* `C` *« à gauche » si* `sign == 1`*, « à droite » sinon.*

Q - 2 : *Tester la fonction en partant d'une liste* `L=[(0,0), (1,0)]`*.*

Pour tracer facilement une courbe à partir d'une liste de tuples contenant les coordonnées cartésiennes des points de la courbe, on souhaite crée une fonction `pts2xyL` qui prend en argument la liste `L` des coordonnées

des points et renvoie deux listes contenant respectivement les abscisses et les ordonnées des points de la liste `L`.

Q - 3 : *Écrire la fonction* `pts2xy`.

Q - 4 : *Tracer les courbes pour obtenir la deuxième image de la partie1.1.*

1.3 Récurrence

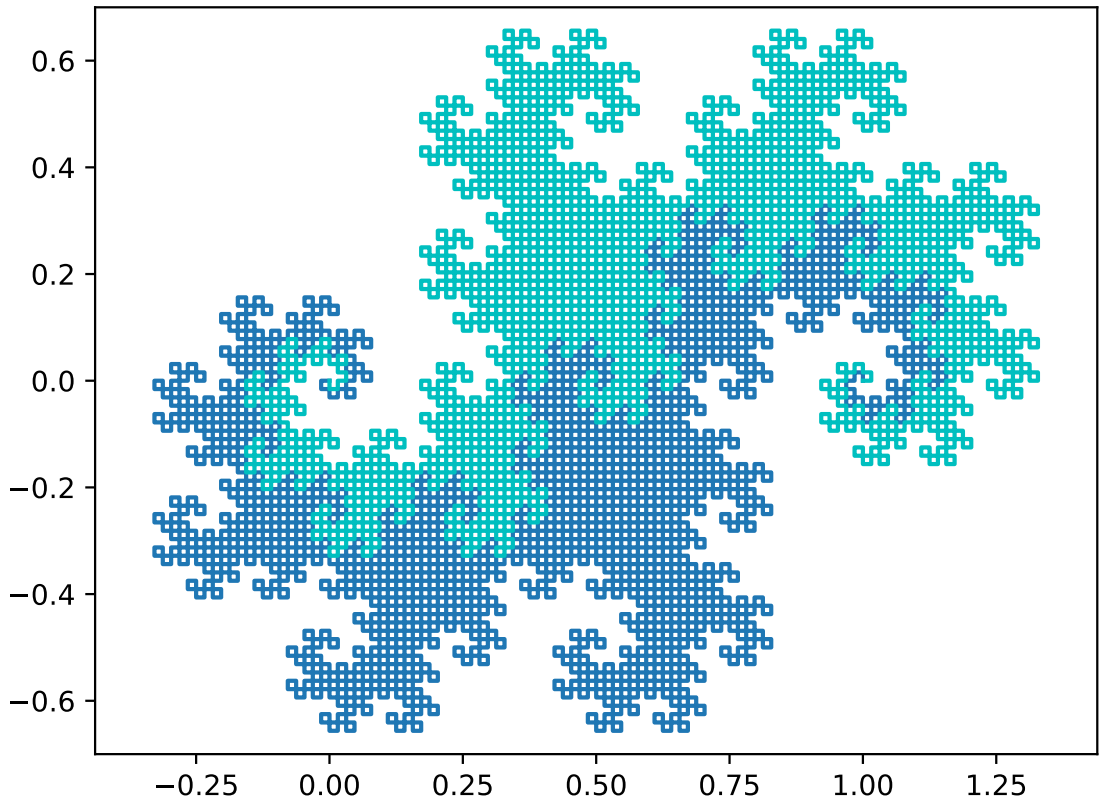
Q - 5 : *Écrire une fonction* `dragon(L, n)` *qui permet d'obtenir la fractale du dragon jusqu'à un niveau de récurrence* `niv` *à partir de la liste de points* `L`. *Le deuxième argument* `n` *correspond au niveau de récurrence de l'appel.*

```
X, Y = (0, 0), (1, 0)
L = [X, Y]
niv = 12
dragon(L,0)
```

1.4 Double dragon

Le double dragon est obtenu en traçant un deuxième dragon à côté du premier en parcourant en sens le segment qui a généré le premier dragon.

Q - 6 : *Écrire une fonction* `doubledragon(L, n)` *qui permet de construire l'image ci-dessous.*

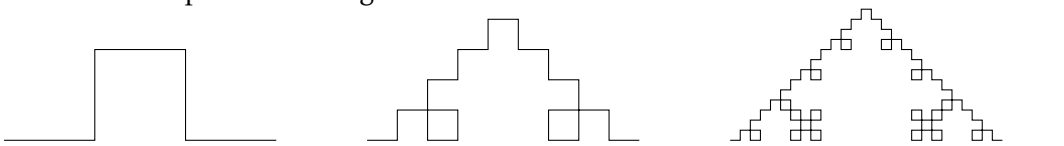


2 Flocon de Koch

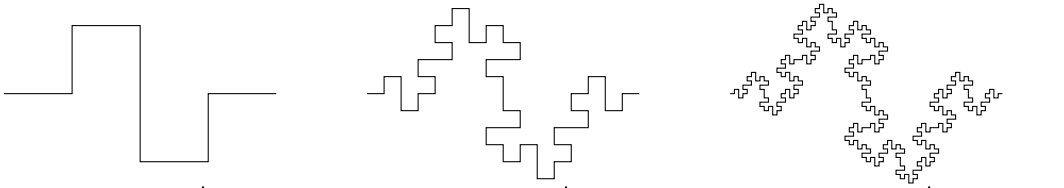
2.1 Motifs

Koch propose plusieurs versions de fractales à partir d'un segment :

Courbe de Koch de type 1 :



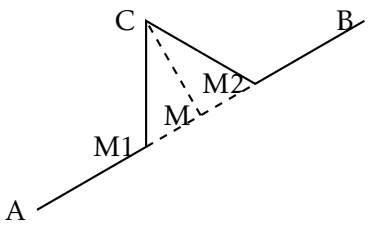
Courbe de Koch de type 2 :



Courbe de Koch de type flocon :

2.2 Point de vue géométrique

On s'intéresse au flocon de Koch :



$$\overrightarrow{AC} = \overrightarrow{AM1} + \overrightarrow{M1C} \quad \text{avec} \quad \overrightarrow{AM} = \frac{1}{2}.\overrightarrow{AB} \quad ; \quad \overrightarrow{AM1} = \frac{1}{3}.\overrightarrow{AB} \quad \text{et} \quad \overrightarrow{AM2} = \frac{2}{3}.\overrightarrow{AB}$$

$$\overrightarrow{AM1} = \left(\frac{x_B - x_A}{3}; \frac{y_B - y_A}{3}\right) \quad \text{et} \quad \overrightarrow{MC} = \left(-\frac{y_B - y_A}{2.\sqrt{3}}; \frac{x_B - x_A}{2.\sqrt{3}}\right)$$

$$C = \left(\frac{x_B + x_A}{2} - \frac{y_B - y_A}{2.\sqrt{3}}; \frac{y_B + y_A}{2} + \frac{x_B - x_A}{2.\sqrt{3}}\right)$$

Les motifs de type 1 et 2 sont un mélange de la courbe du dragon et du flocon.

Q - 7 : *Écrire une fonction* `nouv_pts_flocon(A, B)` *qui prend en argument deux tuples* `A` *et* `B` *contenant les coordonnées cartésiennes des points* `A` *et* `B` *et qui renvoie les coordonnées des points* `M1`, `C` *et* `M2`.

Q - 8 : *Tester la fonction en partant de la liste* `L=[(0,0), (1,0)]` *puis de la liste* `L=[(0,0), (2,1)]`

2.3 Récurrence

Q - 9 : *Écrire une fonction* `flocon(L, n)` *qui permet d'obtenir une courbe de type flocon de Koch jusqu'à un niveau de récurrence* `niv` *à partir de la liste de points* `L`. *Le deuxième argument* `n` *correspond au niveau de récurrence de l'appel.*

Q - 10 : *Appeler la fonction avec une liste de trois points formant un triangle équilatéral.*

3 Arbres personnalisables

3.1 Principe

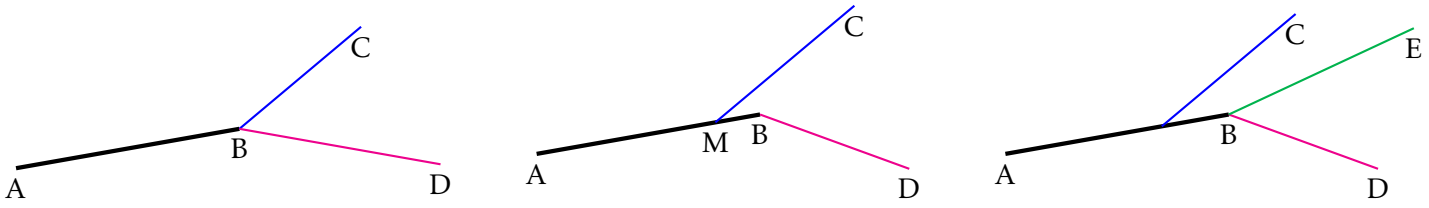
Dans les deux exemples précédents ont ne fait qu'ajouter des points à une courbe : 1 point pour la courbe du dragon et 3 à 6 points pour les courbes de Koch.

Dans l'exemple suivant des arbres, l'approche est plus complexe car il ne s'agit pas d'une courbe mais d'un « arbre » (au sens propre et au sens informatique).

A partir d'un tronc, l'idée est d'ajouter au moins 2 branches en bout de tronc de plus en plus courtes.

3.2 Formes des branches

A partir d'un tronc AB, on peut ajouter deux branches BC et BD en choisissant leurs longueurs et leur angles par rapport au tronc. On pourrait même définir à partir de quelle hauteur du tronc la branche pousse :



3.3 Point de vue géométrique

On s'intéresse au cas à 2 branches dont l'une est n'est pas au sommet du tronc.

On définit 5 ratios $(r, x1, x2, y1, y2)$ tels que :

$\overrightarrow{AM} = r.\overrightarrow{AB} \quad \Rightarrow \quad M = (x_A + r.(x_B - x_A), y_A + r.(y_B - y_A))$ et on pose $\overrightarrow{Perp} = (-(y_B - y_A), x_B - x_A)$

$$\begin{aligned}
\overrightarrow{MC} &= x1.\overrightarrow{AB} + y1.\overrightarrow{Perp} = (x_1.(x_B - x_A) - y1.(y_B - y_A), x1.(y_B - y_A) + y1.(x_B - x_A)) \\
\overrightarrow{BD} &= x2.\overrightarrow{AB} + y2.\overrightarrow{Perp} = (x_2.(x_B - x_A) - y2.(y_B - y_A), x2.(y_B - y_A) + y2.(x_B - x_A)) \\
C &= (x_A + (r + x1).(x_B - x_A) - y1.(y_B - y_A), y_A + (r + x1).(y_B - y_A) + y1.(x_B - x_A)) \\
D &= (x_B + x_2.(x_B - x_A) - y2.(y_B - y_A), y_B + x2.(y_B - y_A) + y2.(x_B - x_A))
\end{aligned}$$

Remarque : on peut aussi choisir $(x1, x2, y1, y2)$ à partir d'un ratio ri et d'un angle θ_i avec $xi = ri.\cos(\theta_i)$ et $yi = ri.\sin(\theta_i)$.

```
r = 0.8
r1 = 0.8
r2 = 0.7
dang1, dang2 = 30, -30
ang1, ang2 = dang1*np.pi/180, dang2*np.pi/180
x1, y1 = r1*np.cos(ang1), r1*np.sin(ang1)
x2, y2 = r2*np.cos(ang2), r2*np.sin(ang2)
```

Q - 11 : *Écrire une fonction* `nouv_pts_arbre(A, B)` *qui prend en argument deux tuples* `A` *et* `B` *contenant les coordonnées cartésiennes des points* `A` *et* `B` *et qui renvoie les coordonnées des points* `M`, `C` *et* `D`.

Remarque : les ratios `r`, `x1`, `x2`, `y1` et `y2` seront des variables globales.

Q - 12 : *Tester la fonction en partant de la liste* `L=[(0,0), (0,1)]`.

3.4 Récurrence

La récurrence permet de tracer l'arbre au fur et à mesure. Pour cela, on crée une fonction `arbre(A, B, n)`.

A chaque niveau `n` :

- on trace le tronc
- si `n < niv`, on calcule les coordonnées des points M, C et D
- on appelle la fonction `arbre` pour les nouvelles branches MC et BD en incrémentant n
- pour plus de fun, on peut se donner une liste de couleurs.

Q - 13 : *Écrire une fonction* `arbre` *pour obtenir l'image ci-dessous.*