Marche aléatoire 2D simple

Dans les modèles les plus simples, on considère un polymère comme un ensemble de segments faisant entre eux un angle quelconque (freely jointed chain). Ce problème est aussi dénommé “marche aléatoire” (random walk) en mathématique ou physique. Il peut aussi rendre compte d'autres phénomènes tel que celui de la diffusion. Après simulation, vous comprendrez pourquoi on appelle “pelote statistique” la configuration d'un polymère en solution.

Le module turtle peut être utilisé pour simuler à 2 dimensions le “parcours” d'une chaîne de N segments de longueur l. L'angle doit être généré par une fonction aléatoire (“random” en anglais).

Le problème pourra aussi être traité en utilisant l'interface Tkinter et les canevas. Voici un programme de base :

<sxh python; title : random_walk_2D-simple.py> #!/usr/bin/env python # -*- coding: utf-8 -*-

Random walk - simulation d'une chaîne de polymère. # Les auto-recouvrements sont possibles

from Tkinter import * from random import randrange from time import sleep

def simu_chain():

```
  can1.delete(ALL)
  long=8
  # les 4 directions sont données dans le sens trigonométrique
  direction=[[long,0],[0,long],[-long,0],[0,-long]]
  xo,yo=250,250
  i,N=0,100
  d_interdit=d=-1  #initialisation
  while i < N:
      #print i
      while d == d_interdit:  #
          d=randrange(4)  #génère un nombre aléatoire entre 0 et 3
      xn,yn=xo+direction[d][0],yo+direction[d][1]
      can1.create_line(xo,yo,xn,yn,width=1,fill='black')
      sleep(0.050)  # attends 50 ms entre chaque segment
      can1.update_idletasks() # pour redessiner à chaque fois le canevas
      xo,yo,i=xn,yn,i+1
      d_interdit=(d+2)%4   # la direction interdite = direction opposée (% =
modulo)
      d=d_interdit
```

programme principal

fen1=Tk() can1=Canvas(fen1,bg='white',height=500, width=500) can1.grid(row=0) # le Canvas occupera le dessus bou1=Button(fen1,text='Quitter',command=fen1.quit) bou1.grid(row=1,sticky=W) #ce bouton sera en dessous à gauche (West) bou2=Button(fen1,text='Générer une chaîne',command=simu_chain) bou2.grid(row=1,sticky=E) #ce bouton sera en dessous à droite (East)

fen1.mainloop()

fen1.destroy()

</sxh>

Améliorations à envisager :

- calcul de grandeurs caractéristiques de la chaîne générée
- statistique sur plusieurs générations
- parcours sur réseau régulier (carré, hexagonal…)
- self-évitement (= self avoiding random walk ou SARW)
- simulation à 3D (géométrie du réseau, représentation)

Références

consulter éventuellement “marche aléatoire” via un moteur de recherche ou une encyclopédie libre.