

marche au hasard à une dimension

On modélise la marche aléatoire selon Ox d'une particule effectuant N pas aléatoires. A intervalles de temps régulier τ , la particule subit un choc à la suite duquel, aléatoirement, elle continue son chemin selon $+Ox$ ou $-Ox$. Entre deux chocs, la particule se déplace donc soit vers la droite soit vers la gauche d'une distance moyenne, appelée libre parcours moyen et notée l .

I. Approche théorique

D'un instant à l'autre, l'abscisse de la particule varie aléatoirement de $+l$ ou $-l$ avec une égale probabilité. On note $p(x, t)$ la probabilité qu'une particule se trouve en x à l'instant t .

1. Exprimer $p(x, t + \tau)$ en fonction de $p(x + l, t)$ et $p(x - l, t)$.

2. On donne le DL à l'ordre 2 en ϵ petit: $p(x + \epsilon, t) = p(x, t) + \epsilon \frac{\partial p}{\partial x}(x, t) + \frac{\epsilon^2}{2} \frac{\partial^2 p}{\partial x^2}(x, t)$.

Ecrire le DL à l'ordre 1 en τ de $p(x, t + \tau)$. Ecrire les DL à l'ordre 2 en l de $p(x + l, t)$ et de $p(x - l, t)$.

3. En déduire que $p(x, t)$ vérifie une équation de la forme $\frac{\partial p}{\partial t} = D \frac{\partial^2 p}{\partial x^2}$. Exprimer D en fonction de l et τ .

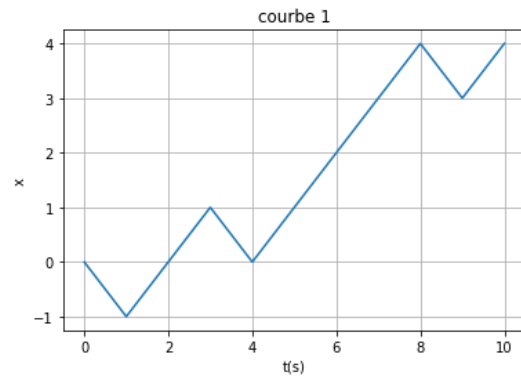
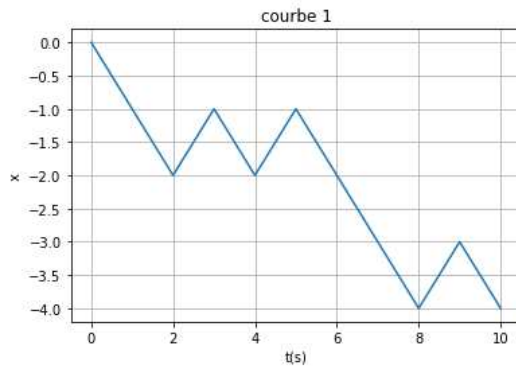
II. Simulation numérique de la marche au hasard

A l'instant $t_0 = 0$, la particule se trouve en $x = 0$. On cherche la position $x(t)$ de la particule aux instants $t_i = \tau, 2\tau, 3\tau, \dots$. On simule la marche au hasard de la particule sous python en utilisant la fonction `random.choice([-l,+l])` qui renvoie aléatoirement la valeur $-l$ ou $+l$.

On donne le code suivant:

```
1 import matplotlib.pyplot as plt
2 import numpy as np
3 import random
4
5 l=1
6 def position(N):
7     poslist=[0]
8     for i in range(N):
9         poslist.append(poslist[i]+random.choice([+l,-l]))
10    return np.array(poslist)
11
12 N=....
13 temps=[j for j in range(N+1)]
14 plt.plot(temps,position(N))
15 plt.title('courbe 1')
16 plt.xlabel('t(s)')
17 plt.ylabel('x')
18 plt.grid()
19 plt.show()
```

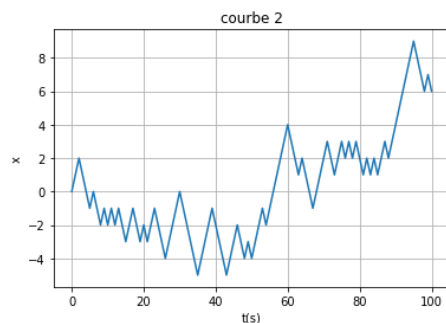
1. On exécute le code deux fois et on obtient:



1.a. Expliquer la ligne 9 et compléter la ligne 12.

1.b. Ecrire le contenu de la liste temps et donner la valeur numérique de τ choisie ici. Ecrire ce que renvoie (numériquement) la fonction `position(N)` pour chacune des courbes obtenues.

1.c. Qu'a-t-on modifié dans le code pour obtenir cette courbe?



2. Les particules, au nombre de N_p , se trouvent initialement à l'origine O des coordonnées, puis chacune d'elles effectue une marche aléatoire de N pas comme dans le paragraphe précédent. On stocke les positions de toutes les particules à tous les instants dans un tableau noté T de $N + 1$ lignes et de N_p colonnes. L'élément de la ligne i et de la colonne j du tableau T se note $[i, j]$, il correspond à l'abscisse $x_j(t_i = i\tau)$ de la particule d'indice j .

$T[i, :]$ désigne la ligne i du tableau T , cette ligne contient les abscisses de toutes les particules à l'instant t_i .
 $T[:, j]$ désigne la colonne j du tableau T , elle contient les abscisses de la particule d'indice j à tous les instants.

2.a. Exemple: on donne le tableau T ci-dessous.

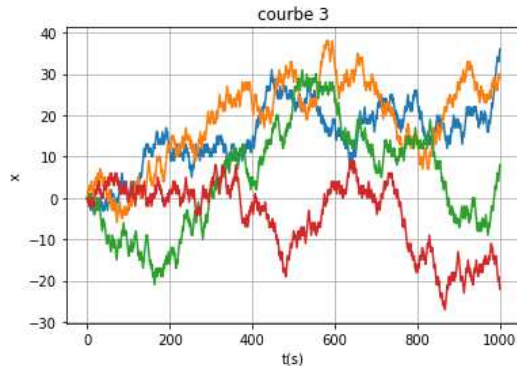
0	0	0
1	1	-1
2	0	0
1	1	-1
2	0	-2

Donner les valeurs de N_p et de N . Donner $x_0(t = 3\tau)$ pour la particule d'indice 0, $x_1(t = 2\tau)$ pour la particule d'indice 1 et $x_2(t = \tau)$ pour la particule d'indice 2. Poser le calcul permettant d'évaluer $\langle x(t = \tau) \rangle$ (valeur moyenne de la position calculée sur l'ensemble des particules à l'instant $t = \tau$). Poser le calcul permettant d'évaluer $\langle x^2(t = 3\tau) \rangle$ (valeur moyenne du carré de la position calculée sur l'ensemble des particules à l'instant $t = 3\tau$).

2.b. On complète le code précédent par

```
30 N=1000
31 temps=[j for j in range(N+1)]
32 for i in range(0,4):
33     plt.plot(temps,position(N))
34     plt.title('courbe 3')
35 plt.xlabel('t(s)')
36 plt.ylabel('x')
37 plt.grid()
38 plt.show()
```

L'exécution du code donne le graphe suivant. Expliquer ce que l'on observe et commenter les courbes.



2.c. On complète le code par:

```
40 Np=1000
41 T=np.zeros((N+1,Np))
42 for i in range(Np):
43     T[:,i]=position(N)
44
45 def f(l):
46     s=0
47     for i in range(len(l)):
48         s=s+l[i]**2
49     return s/len(l)
50
51 def g(l):
52     s=0
53     for i in range(len(l)):
54         s=s+l[i]
55     return s/len(l)
```

Donnée: $T = np.zeros((N + 1, Np))$ crée un tableau contenant $N + 1$ lignes et Np colonnes ne contenant que des zéros.

$T[:, i]$ représente les termes de la colonne i

Que représente le tableau T ?

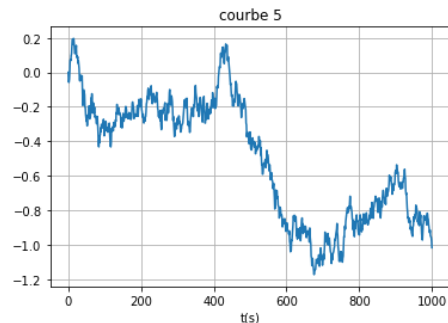
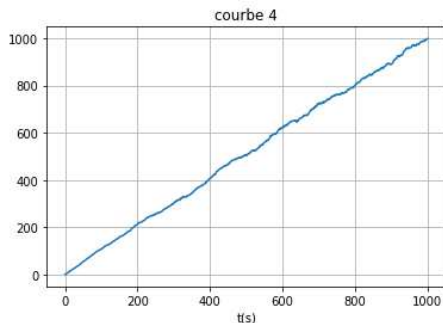
Quelle opération réalise la fonction f ?

Quelle opération réalise la fonction g ?

2.d. On complète une dernière fois le code par (donnée: $T[i, :]$ représente les éléments de la ligne i du tableau):

```
57 l1=[]
58 for i in range(N+1):
59     l1.append(f(T[i,:]))
60 plt.plot(temps,l1)
61 plt.xlabel('t(s)')
62 plt.title('courbe 4')
63 plt.grid()
64 plt.show()
65
66 l2=[]
67 for i in range(N+1):
68     l2.append(g(T[i,:]))
69 plt.plot(temps,l2)
70 plt.xlabel('t(s)')
71 plt.title('courbe 5')
72 plt.grid()
73 plt.show()
```

L'exécution du code donne les courbes suivantes:



Préciser l'ordonnée de la courbe 4 et l'ordonnée de la courbe 5.

Commenter les courbes et conclure.