CN4 – Marche au hasard

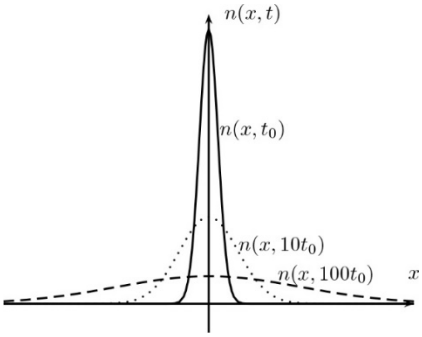
Capacité numérique : à l'aide d'un langage de programmation, simuler la marche au hasard d'un grand nombre de particules à partir d'un centre et caractériser l'étalement spatial de cet ensemble de particules au cours du temps.

I – Simulation à une dimension

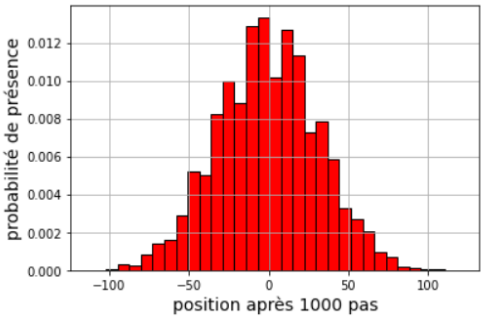
I-1) Problème étudié

On dispose de N_0 atomes dans un tuyau d'axe Ox, de section S, en $x = 0$ à la date $t = 0$. Ils diffusent de façon unidimensionnelle sur l'axe (Ox), dans les deux sens, avec un coefficient de diffusion D.

Les particules diffusent dans le tuyau et vérifie une répartition gaussienne. On souhaite vérifier que l'équation de diffusion qui régit ce phénomène est fortement lié à une marche au hasard où chaque particule à une chance sur deux d'aller vers les x positifs et une chance sur deux d'aller vers les x négatifs.



Répartition des atomes dans un tuyau d'axe (Ox)

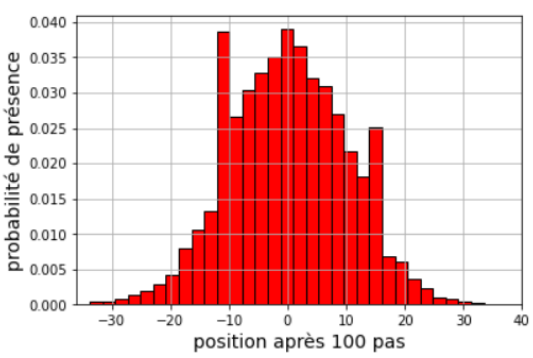


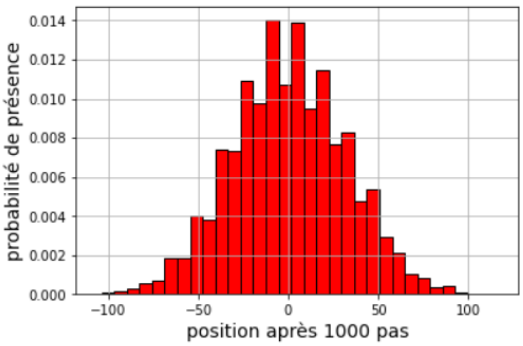
Simulation numérique de 50000 particules après 1000 pas.

I-2) Algorithme utilisé

Pour réaliser la simulation de la marche au hasard, on va utiliser la fonction random.choice qui permet de tirer au hasard un élément d'une liste ici composée de -1 et 1. Ainsi on pourra déterminer où se situe la particule après N_{pas}.

```
import matplotlib.pyplot as plt
import random


def simulation(Npas,Npart):
    L=[-1,1]
    x=0
    y=[]
    for k in range(Npart):
        x=0
        for tirage in range(Npas):
            deplacement=random.choice(L)
            x=x+deplacement
        y.append(x)
    xmin=min(y); xmax=max(y);
    plt.figure(1)
    plt.grid()
    plt.hist(y,range=(xmin,xmax),bins=35,density=True,color='red',edgecolor='black')
    plt.xlim(xmin, xmax)
    plt.xlabel("position après %i pas" %Npas, fontsize=14)
    plt.ylabel(u'probabilité de présence', fontsize=14, rotation=90)
    plt.show()
    return
simulation(100,10000)
simulation(500,10000)
simulation(1000,10000)
```





I-3) Étalement spatial

On cherche à vérifier que l'étalement spatial L est proportionnel à $\sqrt{\tau}$ le temps caractéristique de diffusion. On va faire l'hypothèse que :

- Le nombre de « pas » est proportionnel au temps caractéristique de diffusion
- L'étalement spatial est directement lié à largeur à mi-hauteur des courbes obtenues.

Q1) Relever les largeurs à mi-hauteur pour plusieurs 6 simulations, par exemple sur les graphiques proposés on obtient :

$$\begin{cases} n_1 = 100\ pas \rightarrow L_1 \sim 26\ unités \\ n_2 = 1000\ pas \rightarrow L_2 \sim 85\ unités \end{cases} \Rightarrow \begin{cases} \sqrt{\dfrac{n_2}{n_1}} = 3{,}16 \\ \dfrac{L_2}{L_1} = 3{,}08 \end{cases}$$

Q2) Proposer une méthode de vérification de la loi de proportionnalité.

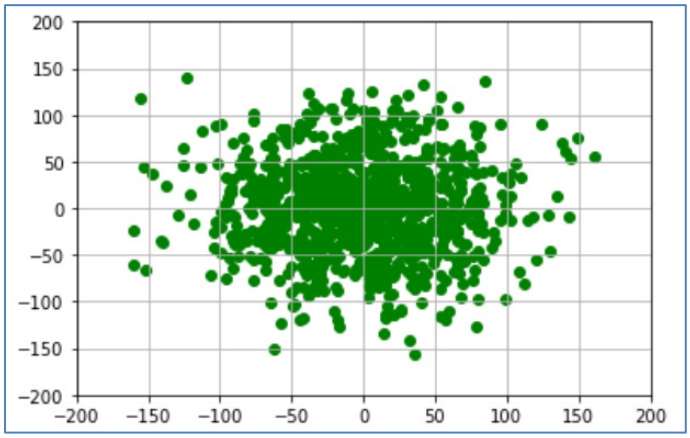
II – Simulation à deux dimensions

II-1) Algorithme utilisé

On désire maintenant représenter la répartition à deux dimensions de N particules. Au début, toutes les particules se situent à t=0, au centre O. Toutes les τ secondes les particules se déplacent d'un pas de longueur fixé dans une direction aléatoire. On va simuler cette marche au hasard et représenter la répartition finale à une date t_f.

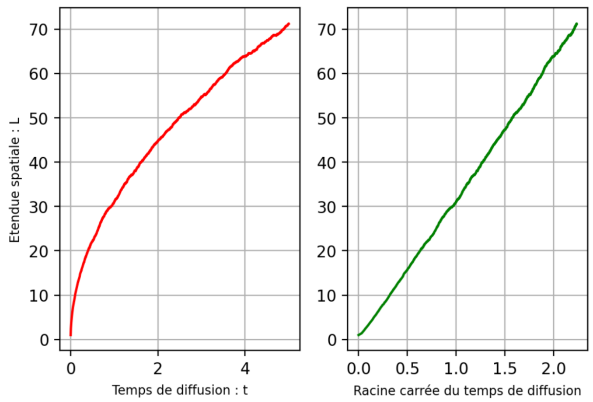
L'utilisateur définit le nombre de particules N_{part} de particules, la date t_f, le pas et les bornes d'affichage. Pour afficher le graphique on utilise l'instruction « plt.scatter » qui donne l'affichage du nuage de points dont x est la liste des abscisses et y celle des ordonnées.

```
from math import *
import random as rd
import matplotlib.pyplot as plt
import numpy as np

def hasard(Npart,tf,pas,borne):
    # initialisation
    Niter=5000
    tau=tf/Niter
    xpart=np.zeros(Npart)
    ypart=np.zeros(Npart)
    # iteration
    for j in range(0,Niter):
        for i in range(0,Npart) :
            # choix aleatoire de la direction de marche
            theta=2*pi*rd.random()
            # deplacement d'un pas dans la direction theta
            xpart[i]=xpart[i]+pas*cos(theta)
            ypart[i]=ypart[i]+pas*sin(theta)
    # affichage nuage de points : scatter
    plt.xlim(-borne,borne)
    plt.ylim(-borne,borne)
    plt.scatter(xpart,ypart,color='green')
    plt.grid(True)
    plt.show()
    return

hasard(1000,5,1,200)
```



II-2) Étendue spatiale

On désire modifier le programme précédent afin de démontrer le lien de proportionnalité entre l'étendue spatiale et le temps de diffusion. Pour cela à chaque itération on doit calculer la distance moyenne parcourue par les particules par rapport à l'origine puis faire une moyenne de celle-ci. On tracera alors la distance $d_{moy} = f(t)$ et on vérifiera graphiquement que c'est bien une fonction proportionnelle à $\sqrt{t}$.



```python
from math import *
import random
import matplotlib.pyplot as plt
import numpy as np
import statistics

def hasard2(Npart,tf,pas,borne):
    # initialisation
    Niter=5000
    tau=tf/Niter
    xpart=np.zeros(1+Npart)
    ypart=np.zeros(1+Npart)
    distancecarree=np.zeros(1+Npart)
    dmoy=np.zeros(1+Niter)
    date=np.zeros(1+Niter)
    rac_date=np.zeros(1+Niter)
    # iteration
    for j in range(0,Niter+1):
        date[j]=j*tau
        rac_date[j]=sqrt(date[j])
        for i in range(0,Npart) :
            # choix aleatoire de la direction de marche
            theta=2*pi*random.random()
            # deplacement d'un pas dans la direction theta
            xpart[i]=xpart[i]+pas*cos(theta)
            ypart[i]=ypart[i]+pas*sin(theta)
            distancecarree[i]=xpart[i]**2+ypart[i]**2
        dmoy[j]=sqrt(statistics.mean(distancecarree))
    # affichage nuage de points : scatter
    plt.clf()
    plt.figure(dpi=200)
    plt.subplot(1,2,1)
    plt.plot(date,dmoy,color='red')
    plt.xlabel("Temps de diffusion : t",fontsize=8)
    plt.ylabel("Etendue spatiale : L",fontsize=8, rotation=90)
    plt.grid(True)
    plt.subplot(1,2,2)
    plt.plot(rac_date,dmoy,color='green')
    plt.grid(True)
    plt.xlabel("Racine carrée du temps de diffusion",fontsize=8)
    #plt.ylabel("Etendue spatiale : L",fontsize=8, rotation=90)
    plt.show()
    return

hasard2(1000,5,1,200)
```

Q3) Expliquer les choix des graphiques représentés. Conclure.