Marche aléatoire et diffusion

Consignes : Les programmes demandés sont a réaliser en Python. N'oubliez de commenter vos codes. Le texte mélange des parties à coder et des explications. Les codes à réaliser sont clairement indiqués dans le texte.
Dans toute la suite on importe les bibliothèques suivantes :

```
import matplotlib.pyplot as plt
import numpy as np
```

1 Marche aléatoire

Nous allons étudier dans cette partie la marche aléatoire dans un plan. La marche aléatoire modélise le déplacement microscopique d'une particule sous l'effet de l'agitation thermique et qui subit régulièrement des collisions. On parle de mouvement Brownien du nom du botaniste Robert Brown qui observa ce mouvement au microscope en 1827 à l'intérieur de grains de pollen. Ce processus est fondamental dans la nature puisqu'il nous permet de comprendre le transport des particules sous l'effet de l'agitation thermique, c'est le phénomène de diffusion. Il permet également de comprendre le processus de transport des photons du cœur des étoiles jusqu'à leur surface.

Nous allons pour l'instant nous intéresser à **un seul marcheur**. Nous pouvons par exemple considérer que notre marcheur est ivre et se déplacement aléatoirement dans toutes les directions.

1.1 Marche aléatoire à 2D

Bloc de code 1 : Un marcheur ivre dans le plan.
Vous devez écrire un code en Python qui permet de tracer le mouvement d'une marche aléatoire de N pas. L'orientation des pas est totalement aléatoire.

1. Vous devez produire le graphe d'une marche aléatoire dans la longueur des pas varie aléatoirement entre 0 et 1 (exemple en figure 1).
2. Vous devez produire le graphe d'une marche aléatoire dont la longueur des pas vaut 1 (exemple en figure 2).

Quelques commandes utiles :
`plt.plot` : Permet de tracer un graphe.
Exemple : `plt.plot(Result2D[:,0], Result2D[:,1], '-')`
`np.random.rand(1)` : Renvoi un nombre aléatoirement tiré dans l'intervalle [0,1).

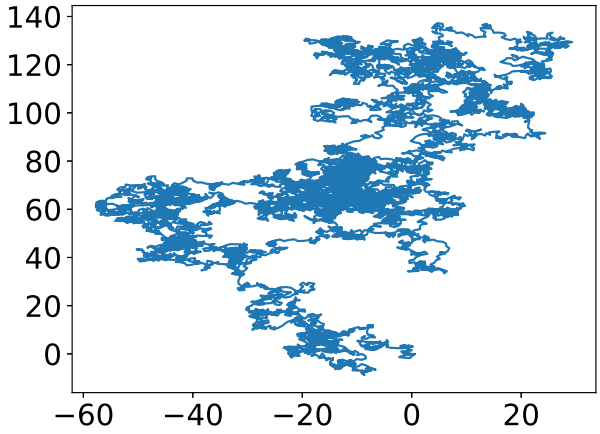

FIGURE 1 – Marche aléatoire de 100000 pas à 2D. L'orientation de chaque pas est aléatoire et la longueur de chaque pas est aléatoirement tirée entre 0 et 1.

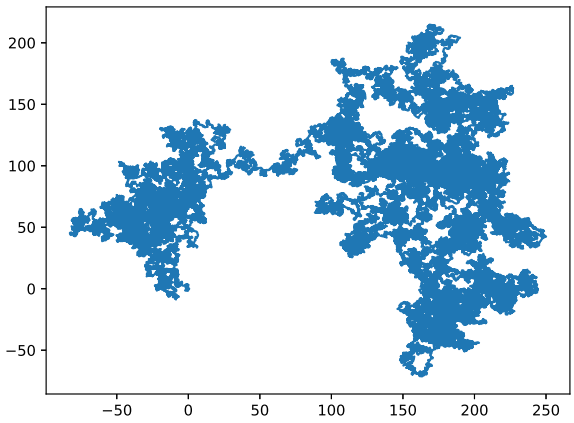

FIGURE 2 – Marche aléatoire de 100000 pas de longueur 1 à 2D. L'orientation de chaque pas est aléatoire.

Utiliser la fonction zoom de la fenêtre pour observer le comportement fractal de la trajectoire. La trajectoire possède une invariance d'échelle, c'est-à-dire qu'un "segment" de la trajectoire a les mêmes propriétés statistiques que la trajectoire entière.

La longueur de la trajectoire dépend de la résolution utilisée pour mesurer cette longueur. Ainsi, si nous voulons mesurer la longueur de la trajectoire pendant un temps T, nous subdivisons cet intervalle en sous intervalles de durée $\tau = \frac{T}{n}$.

La longueur de la trajectoire est alors proportionnelle à $\sqrt{n}$. Ainsi, la longueur mesurée augmente si nous augmentons la résolution de l'appareil de mesure.

1.2 Marche aléatoire à 2D sur réseau

Bloc de code 2 : Un marcheur ivre sur un réseau.
Vous devez écrire un code en Python qui permet de tracer le mouvement d'une marche aléatoire de N pas. Le marcheur peut aller uniquement en diagonal à chaque pas.

1. Vous devez produire le graphe d'une marche aléatoire dont la longueur des pas vaut 1 (exemple en figure 3).

Quelques commandes utiles :
`random.choices` : Permet de tirer aléatoirement un nombre dans un ensemble défini par l'utilisateur.

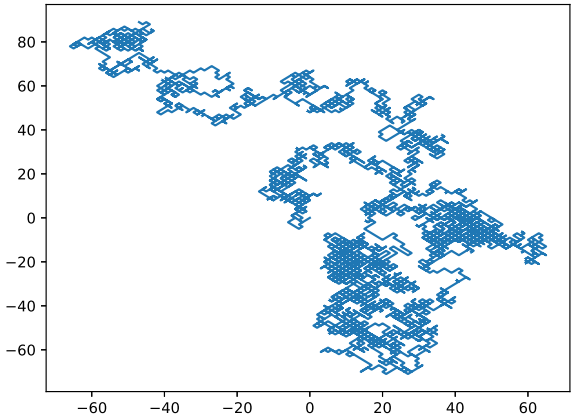

FIGURE 3 – Marche aléatoire de 100000 pas de longueur 1 à 2D. Le marcheur peut aller uniquement en diagonal à chaque pas.

2 Ensemble de marcheurs à 2D

2.1 N_M marcheurs à 2D sur réseau

Bloc de code 3 : N_M marcheurs ivres sur un réseau.
Vous devez écrire un code en Python qui permet de tracer la position finale de N_M marcheurs qui se déplacent aléatoirement de 1000 pas sur un réseau en diagonal. Chaque orientation est équiprobable et chaque pas est de longueur 1.

1. Vous devez produire trois graphes. Un graphe pour $N_M = 100$, un graphe pour $N_M = 1000$ et un graphe pour $N_M = 10000$ (exemple en figure 4).
2. Quel type de symétrie émerge avec l'augmentation du nombre de marcheurs ?

Quelques commandes utiles :
`plt.scatter` : Permet de tracer un nuage de points.

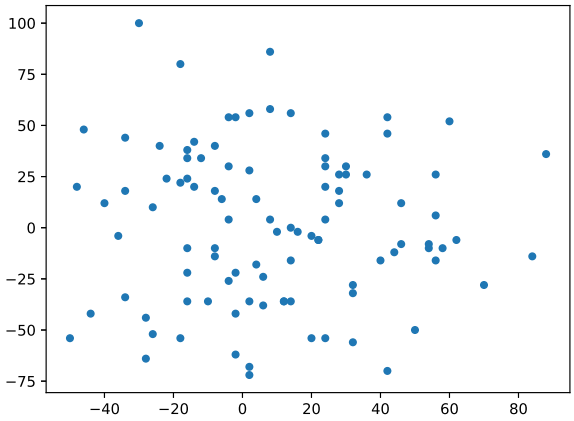

FIGURE 4 – Positions de 100 marcheurs de 1000 pas sur un réseau diagonal à 2D. La longueur de chaque pas est de 1.

2.2 Distance RMS

Nous allons maintenant nous intéresser aux propriétés statistiques du mouvement Brownien. Pour une seule marche aléatoire, la distance qui sépare le point d'arrivée du point de départ est aléatoire mais nous pouvons obtenir une information si nous faisons la moyenne sur un ensemble de marche aléatoire.

Nous nommons $\vec{s}_N$ le vecteur position qui sépare le point d'arrivé du point de départ pour un marcheur. Si nous calculons $< \vec{s}_N >$, nous allons obtenir zéro puisque les marcheurs vont dans des directions aléatoires, nous allons donc calculer $< (\vec{s}_N)^2 >$ où la valeur moyenne est prise sur l'ensemble des marcheurs. Nous notons $\vec{l}_N$ le vecteur déplacement du pas numéro N et nous notons L longueur d'un pas.

1. En écrivant $\vec{s}_N = \vec{s}_{N-1} + \vec{l}_N$, montrer que $< (\vec{s}_N)^2 >=< (\vec{s}_{N-1})^2 > +L^2$.
2. En déduire par récurrence que $< (\vec{s}_N)^2 >= NL^2$.
3. En déduire que la distance RMS (root mean square) a pour expression $\sqrt{< (\vec{s}_N)^2 >} = \sqrt{N}L$.
4. Vérifier qualitativement à l'aide de votre programme que la relation précédente est vérifiée.

3 Ensemble de marcheurs à 1D

Afin d'observer plus finement les propriétés statistiques du mouvement Brownien, nous allons nous intéresser à N_M marcheurs à une 1D. Nous allons ainsi pourvoir facilement tracer l'histogramme des positions finales des N_M marcheurs.

Bloc de code 4 : N_M marcheurs ivres sur un réseau à 1D.
Vous devez écrire un code en Python qui permet de tracer l'histogramme des positions finales de N_M marcheurs qui se déplace aléatoirement de N pas sur un réseau à 1D. Chaque orientation est équiprobable et chaque pas est de longueur 1. Votre code doit également écrire dans la console la valeur moyenne et l'écart type de la distribution.

1. Vous devez produire trois graphes. Un graphe pour $N_M = 100$, un graphe pour $N_M = 1000$ et un graphe pour $N_M = 10000$ avec $N = 1000$ (exemple en figure 5).

2. Vérifier qualitativement que la distribution tend vers une gaussienne.

Quelques commandes utiles :
`plt.hist` : Permet de tracer un histogramme.

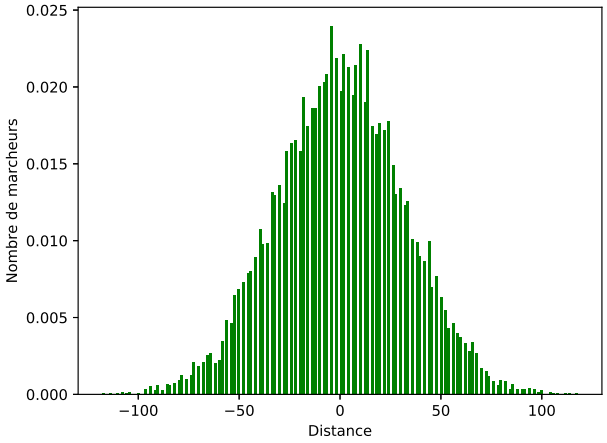

FIGURE 5 – Histogramme des positions finales de 10000 marcheurs de 1000 pas sur un réseau à 1D. La longueur de chaque pas est de 1.

4 Équation de diffusion

4.1 Modélisation de la diffusion

Nous allons maintenant voir que la dynamique d'un ensemble statistique de marcheurs obéit à la loi de la diffusion. Pour modéliser ce processus de diffusion, nous allons considérer un réseau à 1D. Nous nommons i la position d'un point du réseau et $N_M(i,t)$ le nombre de marcheurs au point i et au temps t. Nous nommons p la proportion des N_M marcheurs qui font un pas vers la droite et q la proportion des N_M marcheurs qui font un pas vers la gauche (figure 6).

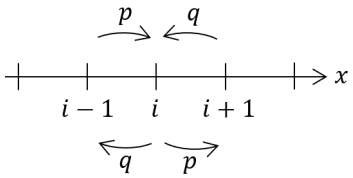

FIGURE 6 – Notations utilisées dans le modèle de diffusion.

L'équation maîtresse qui gouverne l'évolution du nombre de marcheurs en un point quelconque i a donc pour expression :

$$N_M(i,t+1) = N_M(i,t) + qN_M(i+1,t) + pN_M(i-1,t) - qN_M(i,t) - pN_M(i,t)$$

Nous allons ici nous intéresser au cas $p = q$.

Bloc de code 5 : Modélisation de la diffusion.
Vous devez écrire un code en Python qui permet de modéliser l'équation précédente.

1. Vous devez produire un graphe qui montre l'étalement au cours du temps d'une source placée au milieu de l'axe spatiale (exemple en figure 7).

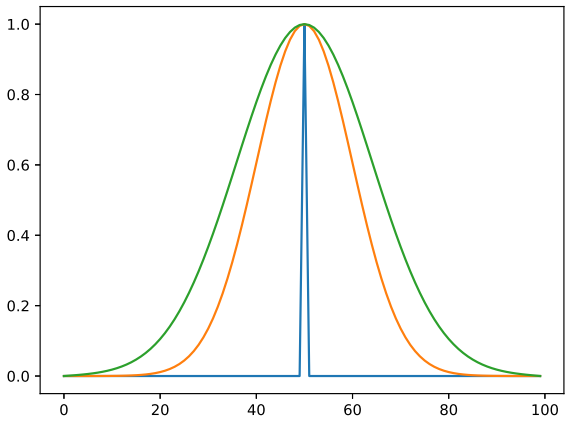

FIGURE 7 – Résultat de l'évolution d'une source placée en $x = 50$ (courbe bleue) au bout de 5000 pas de temps (courbe orange) et 10000 pas de temps (courbe verte) avec $p = q = 0,01$. Les graphes sont normalisés à 1.