MODELING BROWNIAN MOTION

Abdullah Alqahtani[a],Huseen Alenezi [a], Abdulhadi Alqahtani[a]

[a] University of Dayton,Ohio

Introduction

Brownian motion, is the random motion of molecular-sized particles in a fluid like water,milk etc . It results from the stochastic collisions of the particles with the fast-moving molecules in the fluid. They are energized due to the internal thermal energy of the molecules. This motion is quite random.Thus random variable in using numerical methods is used to model and simulate the motion.

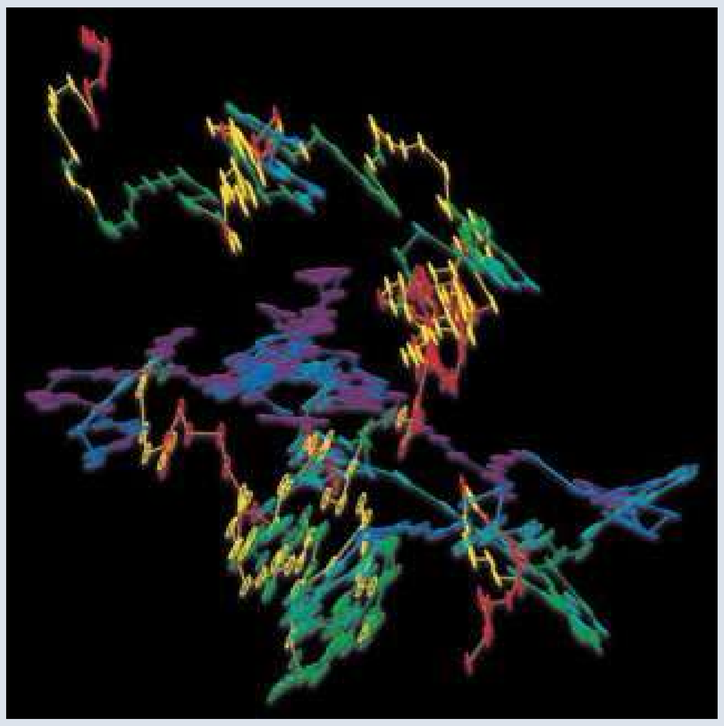


Fig: Tracking of Brownian Motion

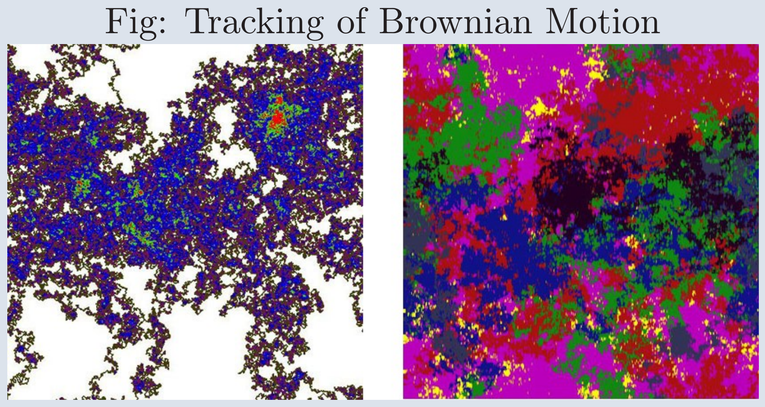


Fig:Two simple random walks in the torus. Points are colored according to occupation measure (left) and hitting time (right).

Methology

First of all, Brownian class is generated in python. Another function of random walk is generated. The diffusion of minute particles suspended in fluid, and other types of diffusion are modeled using the Fokker–Planck and Langevin equations. Process used is Weiner process.

Mathematical Formulation

The key equation that is considered the heart of the modeling of the Brownian motion is given by:

$$W(in) = W(i-1n) + Y_i\sqrt{n}$$

Here "Y" can be a basic stochastic process like Random Walk or sample from a Normal distribution. n is random variable. "i" be the number of steps or number of turns taken for this random process. Usually a lot of of steps are taken. This means usually the value of "i" is very huge.

Python Implementation

import numpy as np from matplotlib import pyplot as plt
class Brownian():
def $_{init}{}_{(self,x0=0):}$
assert (type(x0) == float or type(x0) == int or x0 is None), "Expect a float or None for the initial value"
self.x0 = float(x0)
def gen$_random_walk(self, n_step = 100)$:

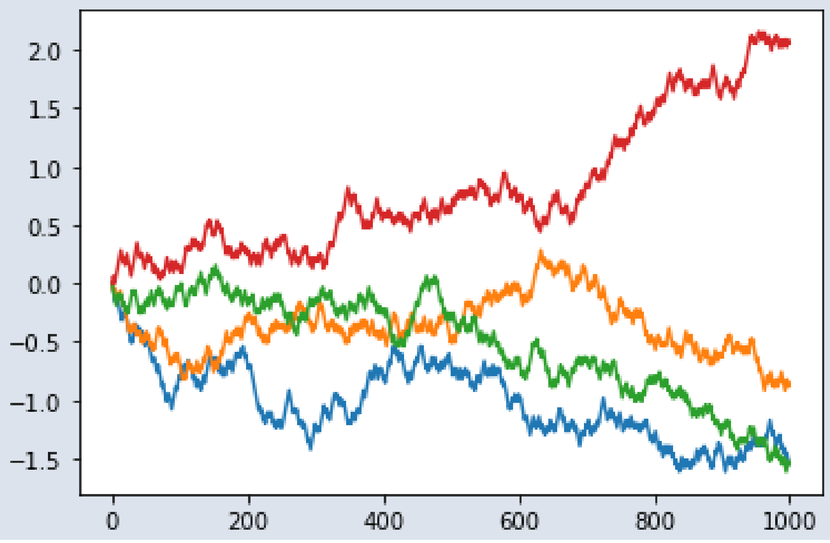
" Warning about the small number of steps" if n$_step$ < 30 : $print("WARNING!The number of steps is small. It may not generate a good stochastic process sequence!")$

w = np.ones(n$_step$) $* self.x0$

for i in range(1, n$_step$) :
$Sampling from the Normal distribution with probability 1/2$
$yi = np.random.choice([1, -1])$
$Weiner process$
$w[i] = w[i-1] + (yi/np.sqrt(n_step))$

return w
b=Brownian()
for i in range(4):
plt.plot(b.gen$_random_walk(1000)$)
$plt.show()$

Results



Conclusion

- Hence the project is implemented and output is taken successfully.

References

https://towardsdatascience.com/brownian-motion-with-python-9083ebc46ff0