

# The Wiener process and the GBM: derivations and simulations

Pietro Colaguori 1936709

November 2023

## 1 Wiener Process

The Wiener process (also known as Standard Brown Motion) is a continuous-time stochastic process named after Norbert Wiener. It serves as a fundamental model for random motion in various fields, including physics, finance, and biology. Let's dive into its key characteristics:

1. **Continuous path:** The Wiener process has a trajectory that is continuous function of time.
2. **Independent increments:** Increments of a Wiener process over non-overlapping time intervals are independent, this implies the Markov properties which makes this process memoryless (just like the Poisson process). Mathematically,  $W(t+u) - W(t)$  is independent of  $W(s), \forall s : s < t$ .
3. **Gaussian distributions:** The increments of the Wiener process are normally distributed, specifically they follow a normal distribution with mean  $\mu = 0$  and standard deviation  $\sigma$  proportional to the length of the time interval  $\Delta t$  considered.
4. **Scaling property:** if you scale time by a factor of  $c$ , the values of the process will be scaled by factor  $\sqrt{c}$ .

Mathematically the Wiener process is denoted by  $W(t)$  and has the property  $W(0) = 0$ .

## 2 Geometric Brownian Motion (GBM)

Geometric Brownian Motion is an extension of the Wiener process and is commonly used to model the evolution of financial assets, such as stock prices. It incorporates the concept of drift and volatility. The stochastic differential equation (SDE) governing GBM is given by:

$$dX_t = \mu X_t dt + \sigma X_t dW_t$$

Where,  $X_t$  is price of the asset at time  $t$ ,  $\mu$  is the drift representing the average rate of return and  $\sigma$  is the volatility, representing the standard deviation of the asset's returns.

Additionally, we note that the GBM exhibits exponential growth due to the stochastic term involving the Wiener process.

### 3 Simulations

I created a Python script that simulates a Brownian motion, of course upon different runs the output changes.

```
1 import numpy as np
2 import streamlit as st
3 import matplotlib.pyplot as plt
4
5 def simulate_wiener_process(num_steps, step_size):
6     # Generate random steps
7     steps = np.random.normal(0, np.sqrt(step_size),
8                               num_steps)
9
10    # Calculate cumulative sum of steps
11    path = np.cumsum(steps)
12
13    return path
14
15 # Parameters
16 num_steps = 1000
17 step_size = 0.1
18
19 # Simulate Wiener process
20 path = simulate_wiener_process(num_steps, step_size)
21
22 # Plot the path
23 st.line_chart(path)
```

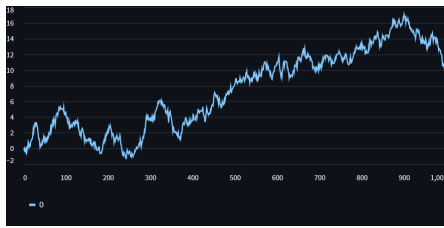


Figure 1: First run.

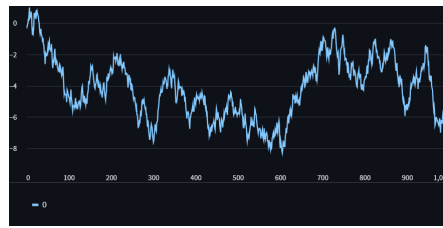


Figure 2: Second run.

Above I show two outputs obtained from two consecutive runs, keeping the number of steps 1000 and the step size equal to 0.1.