The Wiener process and the GBM: derivations and simulations

Pietro Colaguori 1936709

November 2023

1 Wiener Process

The Wiener process 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. **Continous path**: The Wiener process has a trajectory that is continous 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).
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 σ proportional to the length of the time interval Δt 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:

$$dS(t) = \mu S(t)dt + \sigma S(t)dW(t)$$

Where, $S(t)$ is price of the asset at time t, μ is the drift representing the average rate of return and σ 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.

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

def simulate_wiener_process(num_steps, step_size):
    # Generate random steps
    steps = np.random.normal(0, np.sqrt(step_size),
        num_steps)

    # Calculate cumulative sum of steps
    path = np.cumsum(steps)

    return path

# Parameters
num_steps = 1000
step_size = 0.1

# Simulate Wiener process
path = simulate_wiener_process(num_steps, step_size)

# Plot the path
st.line_chart(path)
```

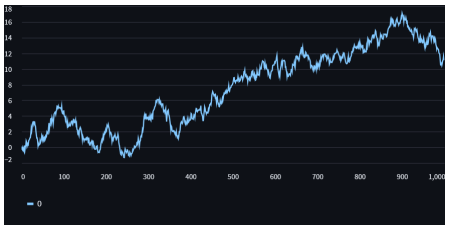

Figure 1: First run.

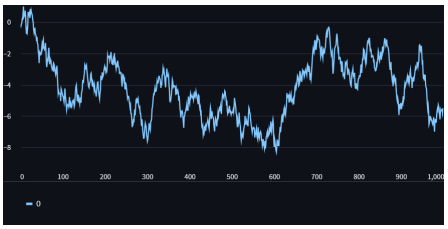

Figure 2: Second run.

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