Class 4 Quiz

Brandon Lewis

September 4, 2025

1 Question 1

The simplest multiplicative (geometric) random walk is defined by

$$x_{n+1} = x_n r_n,$$

where r_n is a positive random factor. This contrasts with the arithmetic random walk, where updates are additive,

$$x_{n+1} = x_n + r_n.$$

Comparison

- **Arithmetic walk:** steps are added (e.g., ± 1). The distribution of x after many steps is approximately normal (bell-shaped) and symmetric about the mean.
- **Geometric walk:** steps are multiplied. After many steps, $\ln(x)$ follows a normal distribution, so x itself follows a log-normal distribution. The result is skewed, with most outcomes near zero and a long tail toward large values.

Python code

Below is a minimal Python script to simulate geometric random walks and plot their probability distribution:

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

N = 1000        # number of steps
trials = 5000   # number of walks
x = np.ones(trials)

for n in range(N):
    r = np.random.choice([0.9, 1.1], size=trials)  # random multiplier
    x *= r

plt.hist(x, bins=100, density=True)
plt.xlabel("Final position x")
plt.ylabel("Probability density")
plt.show()
```

2 Question 2: Why log(x) is normally distributed in GBM

- **Multiplicative process:** $x_{n+1} = x_n r_n$.
- **Take logs:** $\ln(x_{n+1}) = \ln(x_n) + \ln(r_n)$.
- That's just a **sum** of independent random variables.
- By the Central Limit Theorem → the sum tends toward a **normal distribution**.
- Therefore, x itself follows a **log-normal distribution**.

3 Question 3: Langevin approach to stochastic processes

General form:

$$m\frac{dv}{dt} = -\lambda v + \eta + \text{other forces}$$

- $m\frac{dv}{dt}$: inertia (Newton's law).
- $-\lambda v$: damping/friction.
- η: random force (Gaussian noise, models thermal fluctuations).
- Other forces: external fields, interactions, etc.

4 Question 4: Python code to solve 1D Langevin equation (Euler method)

Simplified (set $m = 1$, no external forces):

$$v(t + \delta t) = v(t) - \lambda v(t)\delta t + \eta(t)\delta t \tag{1}$$

$$x(t + \delta t) = x(t) + v(t)\delta t \tag{2}$$

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

dt = 0.01
N = 10000
lambda_ = 1.0
kBT = 1.0
sigma = np.sqrt(2 * lambda_ * kBT / dt)

x = np.zeros(N)
v = np.zeros(N)

for i in range(N-1):
    eta = np.random.normal(0, sigma)
    v[i+1] = v[i] - lambda_ * v[i] * dt + eta * dt
    x[i+1] = x[i] + v[i] * dt

plt.plot(x)
plt.show()
```