$Gabe^2$
GabeL@virginia.edu
https://gabriel-vzh2vs.github.io/SYS3062-Website/
UVA's Systems and Information Engineering

Feburary 17th, 2026

SYS 3062: Lab 4

Simulating Stochastic Processes

The Shift in Thinking

In Labs 2 & 3, we simulated the *final state* of a system (e.g., Portfolio Value at Year 30).
In Lab 4, we simulate the *entire path* of the system over time.

Objectives:

1. **Model Reality:** Use **Geometric Brownian Motion (GBM)** to model stock price movements.
2. **Face Validity:** Compare our simulation against real S&P 500 data (2024).
3. **Pricing Derivatives:** Use these simulated paths to price an **Asian Call Option**.

GabeL@virginia.edu

The Model: Geometric Brownian Motion

We model stock prices (S_t) using a Stochastic Differential Equation (SDE):

The SDE Formula

$$dS_t = \underbrace{\mu S_t dt}_{\text{Drift (Trend)}} + \underbrace{\sigma S_t dW_t}_{\text{Diffusion (Shock)}}$$

Components:

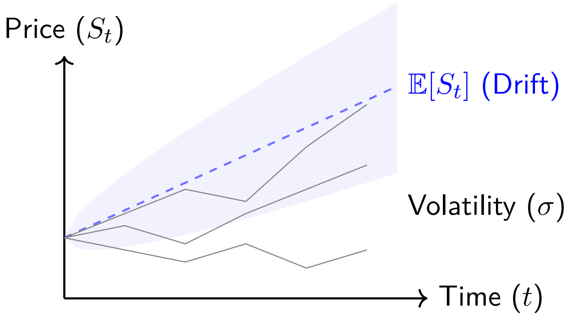
- μ: Expected Rate of Return.
- σ: Volatility (Standard Deviation).
- dW_t: Brownian Motion (Random noise $\sim N(0, dt)$).

Why this model?

- Prices cannot be negative (Log-Normal).
- Volatility is proportional to price (higher price = bigger swings).

Visualizing the Cone of Uncertainty

As time (t) increases, the range of possible outcomes widens. The **Drift** pulls the price up, while **Diffusion** spreads the paths out.



GabeL@virginia.edu

Algorithm: Discretizing the Process

To simulate this on a computer, we solve the SDE using the Euler-Maruyama method (or the exact Log-Normal solution).

Algorithm 1 Simulate GBM Paths

$dt \leftarrow T/N_{steps}$
$S \leftarrow \text{Array}[N_{steps}+1, N_{sims}]$
$S[0] \leftarrow S_0$ ▷ Initial Price
$Z \leftarrow \text{RandomNormal}(0, 1, size = (N_{steps}, N_{sims}))$
for $t = 1$ **to** N_{steps} **do**
    $\text{Drift} \leftarrow (\mu - 0.5\sigma^2) \times dt$
    $\text{Diffusion} \leftarrow \sigma \times \sqrt{dt} \times Z[t-1]$
    $S[t] \leftarrow S[t-1] \times \exp(\text{Drift} + \text{Diffusion})$
end for
return S

Python Implementation

```
def simulate_gbm(S0, mu, sigma, T, n_steps, n_sims=1):
    dt = T / n_steps
    paths = np.zeros((n_steps + 1, n_sims))
    paths[0] = S0

    # Pre-generate Random Shocks
    Z = np.random.normal(0, 1, (n_steps, n_sims))

    for t in range(1, n_steps + 1):
        # Apply Log-Normal Solution
        drift = (mu - 0.5 * sigma**2) * dt
        diffusion = sigma * np.sqrt(dt) * Z[t-1]

        paths[t] = paths[t-1] * np.exp(drift + diffusion)

    return paths
```

GabeL@virginia.edu

Task 1: Face Validity Check

The Test: We verify our model by simulating the S&P 500 using 2024 parameters and overlaying it on actual historical data.

Parameters (2024):

- $S_0 = 4745.20$
- $\mu = 22.1\%$ (Annual Return)
- $\sigma = 13\%$ (Volatility)

Goal

Does the real data fall within the "fan" of our simulated paths? If yes, the model is valid "on its face."

Task 2: Asian Call Option

Definition: An option where the payoff depends on the **Average Price** ($\bar{S}$) over the year, not just the final price.

$$\text{Payoff} = \max(0, \bar{S} - K)$$

The "Model Trust" Question: You must run the simulation twice and compare:

1. **Real-World Model:** Drift $= \mu$ (8%). Represents what we *hope* happens.
2. **Risk-Neutral Model:** Drift $= r$ (3.83%). Represents the theoretical pricing used by banks (discounting risk).

Which premium would you pay? The one based on optimism (μ) or the risk-free rate (r)?

Now, it's your turn to implement Stochastic Processes in Python or using Excel! Good Luck, particularly if you are using Excel.

GabeL@virginia.edu

Switching Gears: Continuous vs. Discrete

Earlier in Lab 4, we modeled **Geometric Brownian Motion**, where the state (S_t) changes continuously every millisecond.

New Challenge: The Poisson Point Process (PPP) Many systems are driven by discrete events (Arrivals), not continuous movement.

- **Counting Process $N(t)$:** The total number of arrivals by time t.
- **Key Property:** It is a "Staircase Function"—it stays flat until an event occurs, then jumps by exactly $+1$.

The Fundamental Difference

- **GBM:** "How much did the price change?" (Magnitude is random).
- **PPP:** "When did the next event happen?" (Timing is random).

Homogeneous Poisson Process

If the average rate of arrivals (λ) is **constant** (e.g., 10 cars/hour all day), the math is simple.

Theory

The time *between* events (Inter-arrival time) follows an **Exponential Distribution**:

$$\Delta t \sim \mathsf{Exp}(\lambda)$$

GabeL@virginia.edu

```
def simulate_homogeneous_ppp(rate, T):
    # 1. Generate Inter-arrival times
    # Scale = 1/lambda because numpy uses beta parameter
    inter_arrivals = np.random.exponential(scale=1/rate, size=N)

    # 2. Cumulative Sum to find actual clock times
    arrival_times = np.cumsum(inter_arrivals)

    return arrival_times[arrival_times <= T]
```

Non-Homogeneous: The "Lunch Rush"

The Problem: Real life is rarely constant. A restaurant has a "Lunch Rush" where $\lambda(t)$ spikes from 5 to 20 customers/hour.
We cannot simply use Exp$(1/\lambda(t))$ because λ changes *during* the wait.

The Solution: Thinning Algorithm (Lewis-Shedler)

1. **Over-generate:** Simulate a homogeneous process at the **Maximum Rate** (λ_{max}).
2. **Filter:** Accept each point with probability $p = \frac{\lambda(t_{current})}{\lambda_{max}}$.

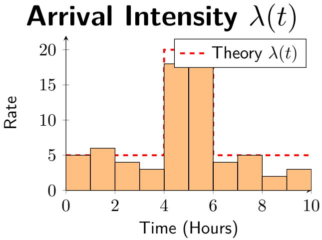
Concept: Generate enough points for the busiest time, then delete (thin) the points during the quiet times.

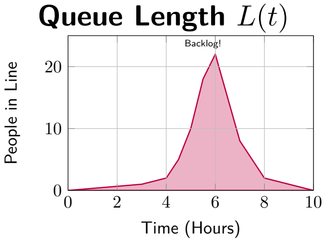
Algorithm 2 Simulate Non-Homogeneous PPP

```
t ← 0
Arrivals ← []
while t < T do
    u ← Random(0, 1)
    t ← t − ln(u)/λ_max                ▷ Next candidate step (Exp)
    if t > T then break
    end if
    D ← Random(0, 1)
    AcceptProb ← λ(t)/λ_max
    if D ≤ AcceptProb then
        Append t to Arrivals           ▷ Event Accepted
    end if
end while
return Arrivals
```


GabeL@virginia.edu

The top plot shows the arrival intensity ($\lambda(t)$). The bottom plot shows the impact on the Queue Length ($L(t)$).





Note: The queue keeps growing even after the rush starts to fade because Arrival Rate (20) > Service Rate (12).

Feedback Form

Throughout the course, I may ask for your feedback on:

- **Course Materials**
- **Lecture & Lab**
- **Assignments**

GabeL@virginia.edu

Action Items

First, I will thank you all again for coming to lab at 5 pm on a Tuesday!

And now, you need to do the following tasks:

- **Submit** your Python file to Gradescope by next week's lab!
- **Read** Prelab 6 for next week!

Next week we are doing hybrid processes!

GabeL@virginia.edu