WIDS Project: Monte Carlo Simulations - Midterm Report Summary

Summary of Weeks 1 & 2

Week 1: NumPy, Vectorization & OOP

The first week focused on establishing a strong foundation in Python programming, specifically targeting efficient numerical computation and modular software design.

Assignment 1.1: The Gambler's Ruin

This assignment introduced the concept of **Vectorized Simulation** to model random walks and variance [**?**].

- **Objective:** Simulate 10,000 gamblers playing a fair coin-flip game simultaneously over 1,000 rounds.
- **The "Iron Rule":** The use of `for` or `while` loops for simulation logic was strictly forbidden. Students were required to use `NumPy` arrays and broadcasting (SIMD operations) to handle the data [**?**].
- **Key Concepts:**
  - **Random Walk:** Modeling financial and physical systems using stochastic processes.
  - **Variance:** Visualizing how wealth distribution spreads over time despite a zero-sum game.
  - **Ruin:** Implementing logic where a gambler stops playing immediately upon reaching a bankroll of $0.

Assignment 1.2: Building Modular Games

This task emphasized **Object-Oriented Programming (OOP)** and clean architecture to prepare for complex simulations [**?**].

- **Objective:** Build a terminal-based game (e.g., Blackjack) using modular design principles.
- **Design Philosophy:** Separation of concerns was critical. Game logic (rules, state) had to be decoupled from the visualization (print statements).
- **Structure:**
  - *Data Model:* Classes for physical components (e.g., `Card`, `Deck`) without rule logic.
  - *Game Engine:* Classes managing flow, state, and rules (e.g., `BlackJack`).
  - *TUI:* Optional creation of an aesthetic Text-Based User Interface using ASCII art.

Week 2: Monte Carlo in Geometry & Finance

The second week applied the vectorization skills to mathematical problems, focusing on integration, convergence, and stochastic financial modeling.

Monte Carlo Integration (Geometry)

The core principle explored was estimating areas by enclosing shapes in a bounding box and calculating the ratio of random points falling inside the shape [**?**].

- **Assignment 2.1 (Estimating π):** Used the ratio of points inside a unit circle vs. a bounding square.

  $$\pi \approx 4 \times \frac{N_{inside}}{N_{total}}$$

  Includes convergence analysis plotting error rates against sample size (N) on a log-log scale.

- **Assignment 2.2 (Estimating e):** Two methods were explored:
  - Calculating the area under $y = 1/x$.
  - A stochastic approach based on the sequence of uniform random variables $u_1 > u_2 > \cdots > u_n$ where the expected value of stopping length n is e.

- **Assignment 2.3 (Modular Shape Estimator):** Refactoring code to be generic, accepting any predicate function (e.g., parabola, Gaussian) to estimate area, reinforcing the DRY (Don't Repeat Yourself) principle.

Optional: Computational Finance (Quant Challenge)

An advanced track focusing on **Geometric Brownian Motion (GBM)** and Option Pricing [**?**].

- **Physics of the Market:** Modeling stock prices S_t using the SDE:

  $$S_t = S_0 \cdot \exp\left(\left(\mu - \frac{\sigma^2}{2}\right)t + \sigma W_t\right)$$

- **European Options:** Pricing Call options by discounting the expected payoff and verifying results against the Black-Scholes analytical formula.

- **American Options:** Introduction to the Longstaff-Schwartz algorithm for pricing options that can be exercised early.