Computational Methods in Finance: Project Report

Pedro R. S. Antunes (Instructor)
Group Members: [Sandip sah, Ugnius Braun, João Relvas]

Submission Deadline: 30 June 2025

Overview

This report documents our solution to the group project assigned in the Computational Methods in Finance course. We implemented, tested, and analyzed stochastic simulation techniques and numerical methods in MATLAB to study pseudo-random number generators, numerical integration of stochastic differential equations (SDEs), and the convergence properties of common numerical schemes. The entire project was designed to be reproducible and all code was developed to automatically generate the figures used throughout the report.

1. Task 1: Random Number Generators

The objective of this task was to implement MATLAB routines to generate pseudo-random numbers from three common distributions and one quasi-random sequence:

- Uniform distribution $U([2,5])$
- Exponential distribution with parameter $\theta = 2$
- Normal distribution $N(0,1)$ using the Box-Muller transform
- Halton sequences in $[0,1]^2$ using prime bases 2 and 3

We used fixed seeds (via `rng`) for reproducibility. Each distribution was visualized using histograms or scatter plots to validate the results.

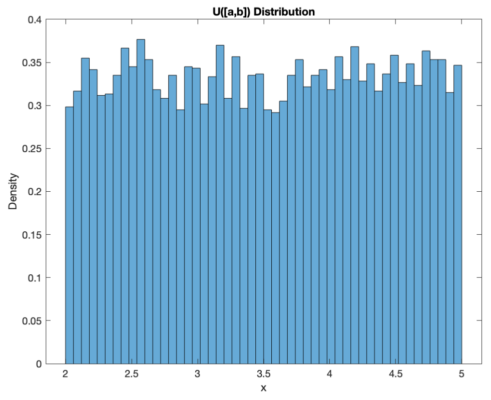
2. Task 2: Estimating Mandelbrot Area

We used both Monte Carlo (MC) and quasi-Monte Carlo (QMC) methods to estimate the area of the Mandelbrot set within the unit square $[0,1] \times [0,1]$. The indicator function of the Mandelbrot region was provided in a MATLAB matrix loaded from `mandelbrot.mat`.

The estimation was performed by checking whether random or Halton points fall inside the fractal region.

- Monte Carlo Estimated Area: **0.32493**
- Quasi-Monte Carlo Estimated Area: **0.32638**

This task demonstrates the effectiveness of QMC methods over standard MC for geometric probability estimation.



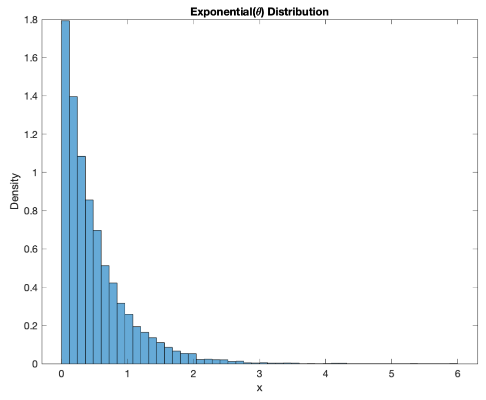
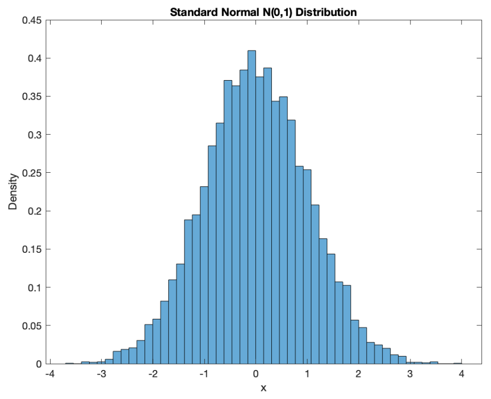


Figure 1: Left: $U([2,5])$ distribution. Right: Exponential($\theta = 2$) distribution



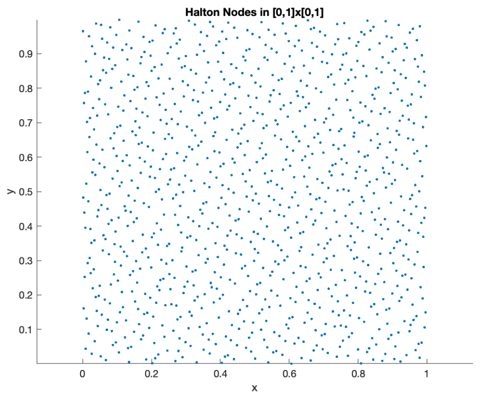


Figure 2: Left: $N(0,1)$ using Box-Muller. Right: Halton nodes in $[0,1]^2$

3. Task 3: Solving SDEs with Euler-Maruyama and Milstein

We implemented two popular numerical methods for solving stochastic differential equations (SDEs):

$$dX(t) = a(t, X(t))\,dt + b(t, X(t))\,dB(t), \quad X(0) = x_0$$

We applied these methods to a linear SDE:

$$dX(t) = 0.5X(t)\,dt + 0.3X(t)\,dB(t)$$

The solver computes the path of $X(t)$ using both the Euler-Maruyama and Milstein methods over a given time discretization, and we plotted the results for visual comparison.

4. Task 4: Geometric Brownian Motion (GBM)

This task involved studying the behavior of numerical solvers when applied to the GBM SDE:

$$dS(t) = \mu S(t)\,dt + \sigma S(t)\,dB(t), \quad S(0) = 1$$

with $\mu = 0.5$, $\sigma = 0.3$, and $T = 1$.

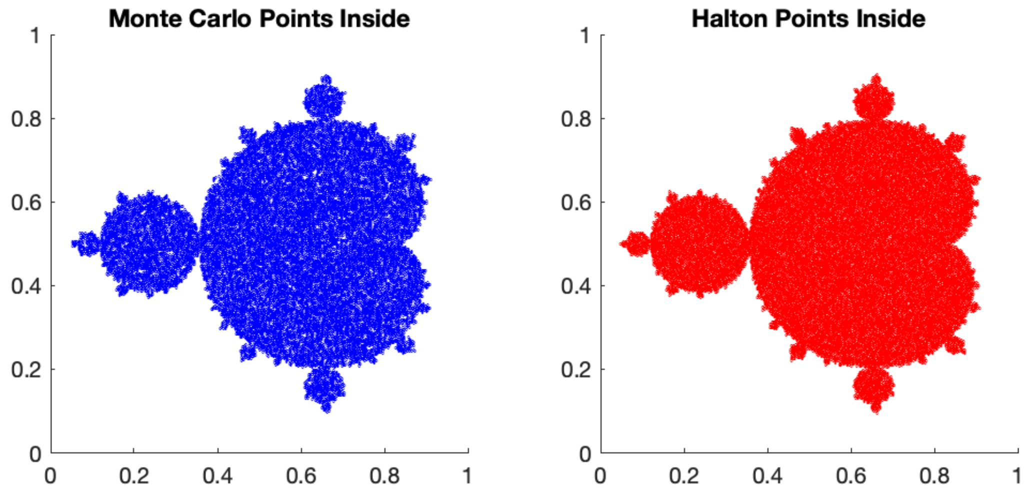


Figure 3: Left: MC points inside Mandelbrot set. Right: QMC (Halton) points

4.1. Task 4(a): Single Path Comparison

We simulated a single realization of $B(t)$ and used it to compare:

- The exact solution
- Euler-Maruyama approximation
- Milstein approximation

4.2. Task 4(b): Convergence Study

We studied the strong and weak convergence behavior of both methods. Using $M = 500{,}000$ simulations for each time step $h = 0.005, 0.0025, 0.00125, 0.000625$, we computed:

- Mean absolute error (strong convergence)
- Difference in means (weak convergence)

The plot confirms that the Milstein method achieves higher order of accuracy, particularly in strong convergence.

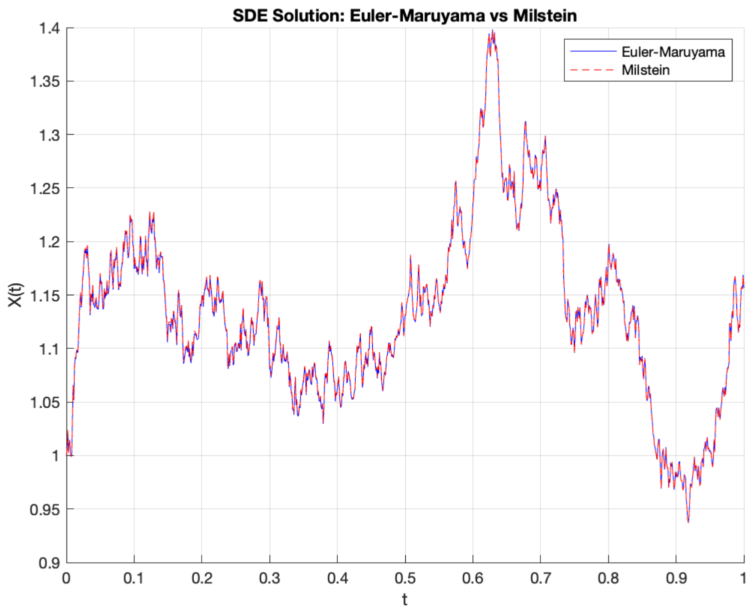


Figure 4: Numerical solution of $X(t)$ using Euler-Maruyama and Milstein methods

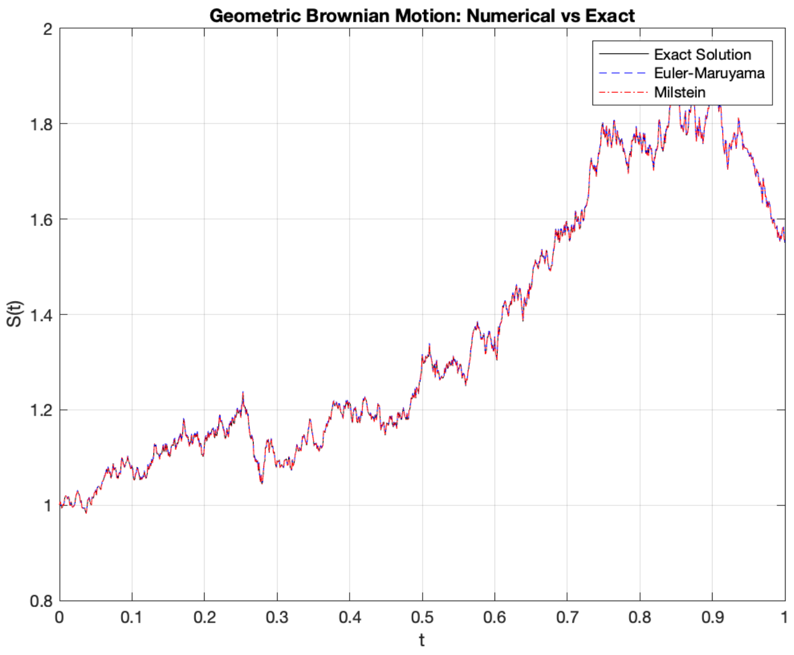


Figure 5: Exact vs numerical solutions for a single GBM path

Reproducibility and Code Summary

All results and figures were generated by MATLAB scripts using fixed random seeds for reproducibility. Executing `main.m` runs all tasks in sequence and generates the `output/` folder with the

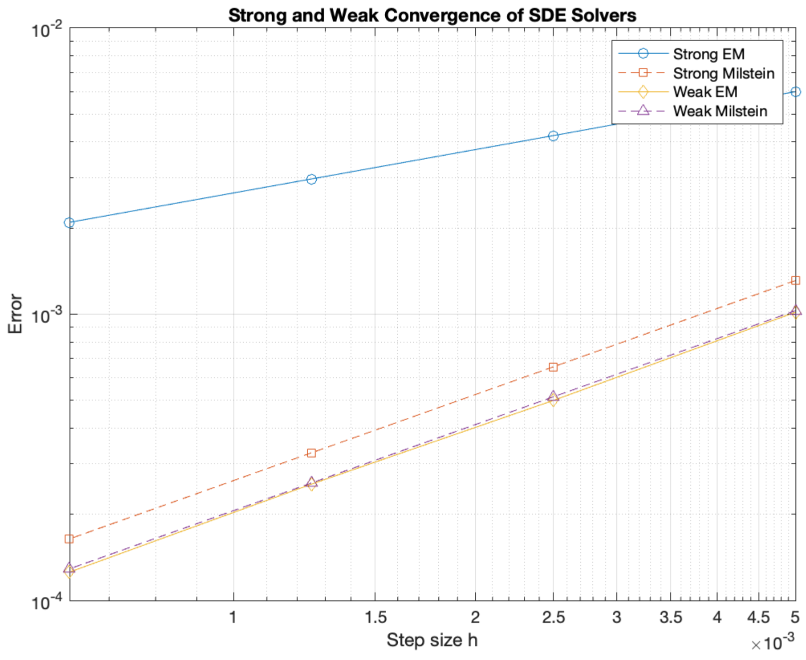


Figure 6: Strong and weak convergence of Euler-Maruyama and Milstein methods

visualizations used in this report.

Code Files Submitted

- `main.m`
- `random_generators.m`
- `estimate_mandelbrot_area.m`
- `sde_solver.m`, `sde_solver_given_path.m`
- `task4a_single_simulation.m`, `task4b_convergence_analysis.m`
- `mandelbrot.mat`
- `output/` folder (all generated figures)