Degree Project in Financial Mathematics, SF291X
Second Cycle, 30 credits

Comparison of Performance for Monte Carlo Methods applied to Barrier Option Pricing

Gustaf Södrén

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ENGINEERING SCIENCES

Author

Gustaf Södrén
KTH Royal Institute of Technology

Place for Project

Stockholm, Sweden

Swedish Title

Jämförelse av Prestanda för Monte Calo Metoder tillämpade på Prissättning av Barriäroptioner

Examiner

Nacira Agram
KTH Royal Institute of Technology

Supervisor

Nacira Agram
KTH Royal Institute of Technology

External Supervisor

Jesper Tidblom
Algorithmica Research AB

Abstract

This thesis compares a set of error reduction techniques such as Quasi Monte Carlo based on Sobol sequences (QMC), Mixed Quasi Monte Carlo (MQMC) and Brownian Bridge (BB) path construction, when they are embedded in the Monte Carlo (MC) framework used by Algorithmica Research AB's Quantlab pricing system.

Up-and-out barrier call options were chosen as the prototype contract. Prices generated with the different schemes were benchmarked against analytic (or high-precision) reference values, using relative error, standard error and average absolute/relative error as performance metrics. Simulations were run under the risk-neutral Geometric Brownian Motion (GBM) model with 1024 time steps and up to roughly 131 000 paths, values chosen to fit Quantlab's native Sobol generator and to reflect real trading usage.

The experiments show that swapping pseudo-random numbers for Sobol' points cuts error sharply by about a factor of 25 at very small path counts for the barrier option. At $N = 1\,023$ paths, the relative error falls from 4.86% with standard MC to just 0.63% with QMC + BB. As the number of paths grows, the gap narrows, yet QMC keeps a clear edge up to roughly 6.5×10^4 paths, where it still delivers five to eight times lower absolute error than MC. Beyond that point all methods converge, but QMC never becomes slower.

The study therefore recommends that Quantlab adopt Sobol/QMC as the default random-number engine and automatically switch to Brownian-Bridge discretisation for barrier options. Doing so can deliver huge speed-ups for complex structures without altering existing payoff code. Future work should extend the benchmark to stochastic-volatility models where similar efficiency gains are expected.

Keywords

Monte Carlo simulation, Quasi Monte Carlo (QMC), Mixed Quasi Monte Carlo (MQMC), Sobol' sequences, Brownian Bridge, Barrier Option, Option pricing, Geometric Brownian Motion, Low-discrepancy sequences

Sammanfattning

Denna avhandling jämför ett antal felreducerande tekniker såsom Quasi Monte Carlo baserat på Sobolsekvenser (QMC), Mixed Quasi Monte Carlo (MQMC) och Brownian Bridge baserad konstruktion när de implementeras i det Monte Carlo (MC) system som används av Algorithmica Research AB, prissättningssystemet Quantlab.

Up-and-out-barriäroptioner (av typen call) valdes som prototypkontrakt. Priser som genererades med de olika metoderna jämfördes med analytiska (eller högprecisions-) referensvärden. Relativt fel, standardfel och genomsnittligt absolut/relativt fel användes som prestandamått. Simulationerna kördes under den riskneutrala modellen Geometrisk Brownsk Rörelse (GBM) med 1024 tidsteg och upp till cirka 131 000 simuleringar vilket är värden som är anpassade till Quantlabs inbyggda Sobol generator och speglar en praktisk marknadstillämpning.

Experimenten visar att ersättning av pseudorandomiserade tal med Sobol punkter drastiskt minskar felen med ungefär en faktor 25 vid mycket få simuleringar för barriäroptionen. Vid $N = 1023$ simuleringar sjunker det relativa felet från 4.86% med standard MC till endast 0.63% med QMC + BB. När antalet samples ökar minskar gapet och resultatet blir mer jämnt, men QMC behåller ett tydligt försprång upp till ungefär 6.5×10^4 simuleringar, där det fortfarande ger fem till åtta gånger lägre absolutfel än MC. Därefter konvergerar alla metoder, men QMC blir aldrig långsammare.

Studien rekommenderar därför att Quantlab använder Sobol/QMC som standard generator för slumptal och automatiskt aktiverar Brownian Bridge diskretisering för barriäroptioner. Detta kan ge avsevärt snabbare beräkningar för komplexa strukturer utan att befintlig payoff kod behöver ändras. Framtida arbete bör utvidga resultaten till stokastiska volatilitetsmodeller, där liknande effektivitetsvinster väntas.

Nyckelord

Monte Carlo-simulering, Quasi Monte Carlo (QMC), Slumpmässig Quasi-Monte Carlo (MQMC), Sobol'-sekvenser, Brownian Bridge, Barriäroption, Optionsprissättning, Geometrisk Brownsk Rörelse, Låg-diskrepanssekvenser

Acknowledgements

Firstly, I would like to thank my supervisor Jesper Tidblom at Algorithmica Research AB for taking on this project and giving great inputs regarding code, theory and final report. I would also like to thank the rest of the colleagues at Algorithmica Research AB for their support. Lastly, I would like to thank my supervisor and examiner Nacira Agram from the department of Mathematics at KTH Royal Institute of Technology.

Contents

List of Tables

List of Figures

1. Introduction

1.1 Background

When pricing complex financial derivatives, financial institutions often rely on Monte Carlo methods because these methods handle path-dependent features and high-dimensional problems more flexibly than many alternatives. However, a recurring challenge is the large error spread typically associated with straightforward MC simulations, requiring a large number of simulated paths to achieve sufficient accuracy. This computational intensity becomes especially difficult for intricate option payoffs such as barrier, asian, and lookback options that demand detailed tracking of the underlying asset's path [8].

Monte Carlo methods are a broad class of computational algorithms that rely on repeated random sampling to obtain numerical results. The key idea is to use randomness to explore the behavior of complex systems or mathematical models when direct analytical solutions are difficult or impossible. Since their inception in the mid-20th century, MC methods have spread into an impressive array of scientific, engineering, and business applications. They are now used to simulate everything from subatomic particle interactions to financial portfolio outcomes, giving researchers a versatile toolkit for understanding uncertainty and risk in complex models [26].

In collaboration with Algorithmica Research AB, this project specifically investigates how different MC techniques can reduce the error in option pricing, thus lowering overall computational requirements. Algorithmica's proprietary environment, Quantlab, offers built-in functionalities for path generation, random number streams, and direct charting capabilities. These features streamline experimental comparisons between standard MC methods and variants incorporating QMC sequences or Brownian Bridge constructions. By comparing the relative errors and computational performance under various parameter settings such as changes in the number of discretization steps, paths, or correlation structures one gains deeper insights into the practical trade-offs of each method.

Ultimately, the goal is twofold: first, to systematically evaluate how each error reduction technique behaves on a Barrier Option, and second, to identify a robust approach that practitioners can adopt for lower error without significantly increasing implementation complexity. Through empirical results and quantitative analysis, the thesis aims to illustrate which MC adaptations best balance accuracy and speed within a real-world financial engineering context.

1.2 Problem and Purpose

Algorithmica Research AB currently uses MC simulations in their product Quantlab to price various financial derivatives. However, standard MC methods often suffer from high variance, which in turn demands a large number of simulated paths to reach acceptable accuracy. This can become computationally intensive, especially when dealing with higher-dimensional options or complex path dependencies (e.g., Barrier or Asian features).

The problem, therefore, is to identify and evaluate error reduction techniques such as QMC and BB to mitigate this computational problem while maintaining or improving pricing precision. By systematically comparing these methods in Quantlab, one can discover which

approach or combination of approaches yields the best trade-off between accuracy and runtime. Furthermore, since the dimension is limited, one usually needs to mix MC and QMC to achive better accuracy.

The primary purpose of this study is thus to develop and test practical Monte Carlo enhancements that reduce the error in option pricing scenarios. The anticipated outcome is a set of recommendations for Algorithmica on how best to incorporate these error reduction techniques into Quantlab, ultimately providing faster and more reliable derivative pricing to their end users.

1.3 Research Questions

- *Which Monte-Carlo Method gets the lowest error when pricing Barrier options?*
- *Which method/mix should you use?*
- *How low is it possible to get the error?*

1.4 Scope

The primary goal of this thesis is to investigate and compare how different Monte Carlo methods, specifically standard Monte Carlo (MC), Quasi Monte Carlo (QMC), Mixed Quasi Monte Carlo (MQMC) and Brownian Bridge (BB) perform when pricing a Barrier Option. Throughout the project, the focus is on identifying which methods offer the most significant error reduction relative to their computational cost, thereby providing a practical recommendation for improving efficiency in existing pricing workflows at Algorithmica Research AB.

In meeting this goal, the thesis includes:

- A theoretical overview of Monte Carlo simulation under the risk-neutral measure, covering the law of large numbers, the central limit theorem, and basics of generating (pseudo)random numbers.
- An implementation component within Algorithmica's Quantlab environment, where each method is coded and tested across sample options to evaluate convergence properties, required sample sizes, and computational runtimes.
- A comparative analysis, presenting quantitative results (relative, standard and average absolute/relative errors) to determine whether QMC + BB or MQMC + BB yield notable accuracy gains versus standard MC.

By confining attention to this set of models and option types, the scope remains both feasible within the given time frame and sufficiently comprehensive to generate meaningful insights.

1.5 Limitations

The project primarily examines Geometric Brownian Motion (GBM) under the risk-neutral measure, along with Brownian Bridge and Quasi Monte Carlo variants. More advanced stochastic volatility models (e.g., Heston) are mentioned for context but not exhaustively implemented or calibrated.

Furthermore, real-world market frictions, transaction costs, and liquidity constraints are not modeled. All simulations assume idealized conditions, including constant interest rates and no market impact, so the final outcomes may not fully capture real trading dynamics.

Finally, due to the limited timeframe of the degree project, the number of tested options, underlying models, and parameter calibrations remains finite. The main aim is to identify general performance trends among a focused option and error reduction strategies rather than to provide an exhaustive market study.

1.6 Summary of thesis

Chapter 1 has introduced the background, problem and purpose of the study as well as the research questions. Also, the scope and limitations were presented. In Chapter 2 the theoretical background necessary to understand the study is presented and the use of it in option pricing. In Chapter 3 the research design which allows the research questions to be answered is presented. This includes the method. In Chapter 4 the results of the study are presented. Finally in Chapter 5, the results are analyzed and conclusions are drawn. The results indicate that error reduction methods reduce the computational expense and error by a great deal.

2. Theoretical Background

In this chapter, a detailed description about relative background of the degree project is presented to better understand results, discussions and conclusions.

2.1 Principles of Monte Carlo

Monte Carlo methods are used in a wide range of fields to estimate quantities that may be too difficult or complex to solve analytically. In a financial context, for example, one often wants to model stock price paths in order to price derivatives (e.g. Asian or European options). A common modeling approach is to assume a stochastic process for the asset price, such as Geometric Brownian Motion, and then sample many paths from this process to compute the expected payoff of a derivative under appropriate pricing assumptions [8].

Generally, Monte Carlo methods estimate an expectation by simulating N random samples and computing an average. The simplest setup aims to evaluate

$$\mathbb{E}_{\psi(\mathbf{x})}[\, f(\mathbf{x})\,] \;=\; \int f(\mathbf{x})\,\psi(\mathbf{x})\,\mathrm{d}\mathbf{x},$$

where $\mathbf{x} \in \mathbb{R}^n$, f is the function of interest, and ψ is the probability density. In practice, we approximate this integral by generating a large number of independent draws $\mathbf{x}_1, \ldots, \mathbf{x}_N \sim \psi$ and taking

$$\frac{1}{N}\sum_{i=1}^{N} f(\mathbf{x}_i).$$

As N grows, this sample average converges to the true value thanks to fundamental results such as the Law of Large Numbers (LLN) and Central Limit Theorem (CLT). In fact, by the Central Limit Theorem, the Monte Carlo estimator is approximately normally distributed around the true value for large N, with a standard deviation (standard error) on the order of $1/\sqrt{N}$. In other words, if $\sigma^2 = \mathrm{Var}[f(X)]$ under the sampling distribution, then the sample mean has standard error $\sigma/\sqrt{N}$. This implies that to reduce the error (standard deviation of the estimate) by a factor of 2, one requires roughly 4 times more samples [13].

In quantitative finance, one typically works under the *risk-neutral measure* $\mathbb{Q}$, meaning the drift of the asset process is replaced by the risk-free rate r. For example, a GBM can be written as

$$\mathrm{d}S(t) \;=\; r\,S(t)\,\mathrm{d}t \;+\; \sigma\,S(t)\,\mathrm{d}W(t),$$

where $W(t)$ is a standard Brownian Motion, σ is the volatility, and r is the risk-free rate. Under this assumption, the fair price of a derivative with payoff V_T at maturity T is

$$V_0 \;=\; \mathrm{e}^{-rT}\,\mathbb{E}^{\mathbb{Q}}[\, V_T\,].$$

Hence, using Monte Carlo under this measure involves simulating multiple GBM paths $S(t)$ and then averaging the discounted payoffs. Crucially, these simulations rely on the assumption that draws are independent and identically distributed (i.i.d.) and that we can generate large samples to apply the LLN and CLT in practice [8].

2.1.1 Geometric Brownian Motion (GBM)

GBM describes the evolution of an asset price $S(t)$ over time according to the stochastic differential equation (SDE) [8]:

$$dS(t) = \mu S(t)dt + \sigma S(t)dW(t), \tag{2.1}$$

where $S(t)$ is the asset price at time t, μ is the drift (expected return), σ is the volatility and $W(t)$ is a standard Brownian motion. Solving the SDE using Itô's Lemma leads to the explicit solution:

$$S(t) = S(0) \exp\left[\left(\mu - \frac{1}{2}\sigma^2\right) t + \sigma W(t)\right]. \tag{2.2}$$

From this, it can be shown that $S(t)$ is lognormally distributed:

$$\log S(t) \sim \mathcal{N}\left(\log S(0) + \left(\mu - \frac{1}{2}\sigma^2\right) t, \sigma^2 t\right). \tag{2.3}$$

The discrete time approximation is then:

$$S_{t+\Delta t} = S_t \exp\left[\left(\mu - \frac{1}{2}\sigma^2\right) \Delta t + \sigma\sqrt{\Delta t} Z\right], \tag{2.4}$$

where $Z \sim \mathcal{N}(0, 1)$ is a standard normal variable.

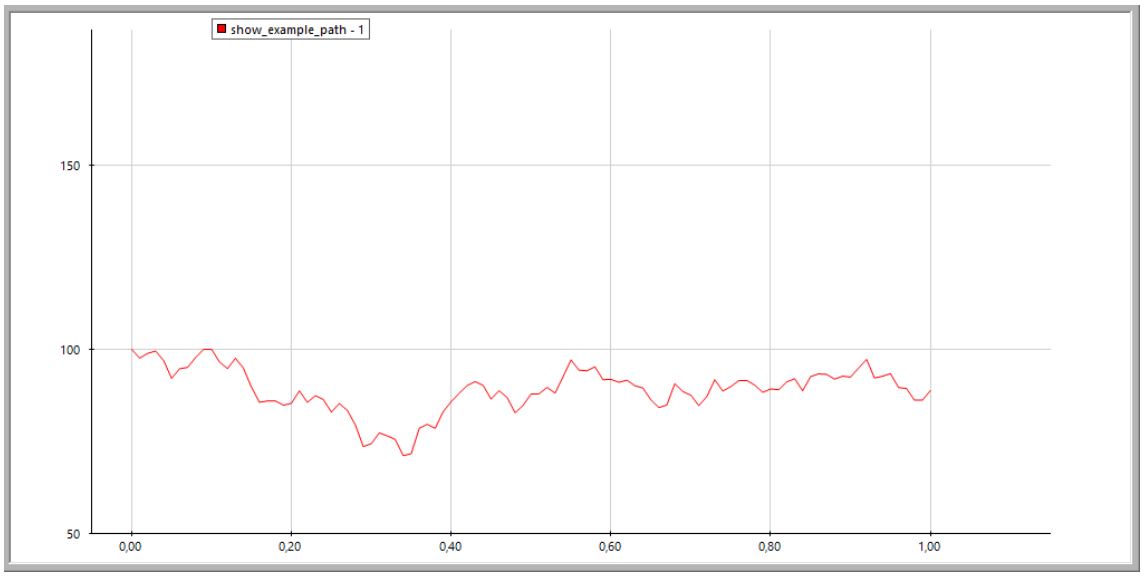


Figure 2.1: Example of GBM with parameters $r = 0.02$, $\sigma = 0.3$, $T = 1$, and $\Delta t = \frac{1}{100}$.

2.1.2 Risk-Free Measure

The risk-free measure is a probability measure in which the expected return of all tradable assets equals the risk-free rate. It plays a underlying role in asset pricing. Let $S(t)$ represent the price of a financial asset. Under the real-world probability measure $\mathbb{P}$, its dynamics follow the stochastic differential equation (SDE):

$$dS(t) = \mu S(t)dt + \sigma S(t)dW_{\mathbb{P}}(t), \tag{2.5}$$

where μ is the drift, σ is the volatility, and $W_{\mathbb{P}}(t)$ is a standard Brownian Motion under $\mathbb{P}$. Under the risk-neutral measure $\mathbb{Q}$, the drift term μ is replaced by the risk-free rate r, leading to the changed SDE:

$$dS(t) = rS(t)dt + \sigma S(t)dW_{\mathbb{Q}}(t), \tag{2.6}$$

where $W_{\mathbb{Q}}(t)$ is a Brownian motion under $\mathbb{Q}$. This transformation allows us to calculate the value of financial derivatives by estimating their expected future payout and then adjusting for time using the risk-free interest rate [8].

The existence of a risk-neutral measure is guaranteed by the Fundamental Theorem of Asset Pricing (FTAP), which states that a market is arbitrage-free if and only if there exists an equivalent martingale measure under which discounted asset prices are martingales [10].

Using the risk-neutral measure, the price V_0 of a derivative with payoff V_T at time T is given by:

$$V_0 = e^{-rT}\mathbb{E}^{\mathbb{Q}}[V_T]. \tag{2.7}$$

This equation states that the fair price of the derivative is the discounted expected value of its future payoff under $\mathbb{Q}$. Monte Carlo simulations often rely on this pricing principle by generating sample paths under the risk-neutral measure[8].

The transformation from the real-world measure $\mathbb{P}$ to the risk-neutral measure $\mathbb{Q}$ is facilitated by Girsanov's theorem, which modifies the drift of Brownian Motion. The Radon-Nikodym derivative defines this measure change as:

$$\frac{d\mathbb{Q}}{d\mathbb{P}} = \exp\left(-\int_0^T \theta_t dW_{\mathbb{P}}(t) - \frac{1}{2}\int_0^T \theta_t^2 dt\right), \tag{2.8}$$

where θ_t is the market price of risk[8].

2.1.3 Law of Large Numbers and Central Limit Theorem

Both the Law of Large Numbers (LLN) and Central Limit Theorem (CLT) are important assumptions when working with Monte Carlo simulations.

The Law of Large Numbers states that as the number of simulations increases, the Monte Carlo estimates converges to the true expected value. I.e when evaluating the function f at n of these random points and averaging the results produces the Monte Carlo estimate.

$$\hat{\alpha}_n = \frac{1}{n}\sum_{i=1}^{n} f(U_i). \tag{2.9}$$

If f is indeed integrable over $[0, 1]$, then, by the strong law of large numbers,

$$\hat{\alpha}_n \to \alpha \quad \text{with probability 1 as } n \to \infty. \tag{2.10}$$

Furthermore, the Central Limit Theorem (CLT) states that if one take a sufficiently large number of independent and identically distributed (i.i.d.) random variables, their sample mean will approximate a normal distribution, regardless of the original distribution of the variables.

Mathematically, if $X_1, X_2, \ldots, X_n$ are i.i.d. random variables with mean μ and variance σ^2, then the sample mean:

$$\bar{X}_n = \frac{1}{n} \sum_{i=1}^{n} X_i \tag{2.11}$$

follows approximately a normal distribution for large n:

$$\bar{X}_n \sim \mathcal{N}\left(\mu, \frac{\sigma^2}{n}\right). \tag{2.12}$$

This implies that the standard error of the sample mean decreases as $O(n^{-1/2})$, meaning that increasing the number of samples improves the estimate.

2.1.4 Expected Value

In probability theory, the expected value of a random variable is the probability-weighted average of all possible values it can take. For a continuous random variable X with density $\psi(x)$, the expected value of a function $f(X)$ is defined as:

$$\mathbb{E}[f(X)] = \int_{-\infty}^{\infty} f(x)\, \psi(x)\, dx,$$

provided this integral converges. For a discrete random variable, the expected value is the sum of $f(x)$ over all possible outcomes weighted by their probabilities. The expected value represents the long-run average outcome of X. Monte Carlo simulation essentially estimates such an expectation by averaging samples drawn from the distribution of X [28].

2.1.5 Generic Monte Carlo Algorithm

The basic steps for Monte Carlo pricing include:

- Simulate N paths of the underlying asset price using the stochastic differential equation (SDE) that models the asset dynamics. For a Geometric Brownian Motion (GBM), the SDE is:

  $$dS(t) = rS(t)dt + \sigma S(t)dW(t), \tag{2.13}$$

  where $dW(t)$ represents the Wiener process.
- Calculate the payoff of the option for each path at maturity.
- Discount the payoff to the present value using the risk-free rate r.
- Compute the average of the discounted payoffs to estimate the option price.

2.2 Introduction to Option Pricing

Option pricing is a key concept for determining the fair price of a financial instrument, such as a put or a call option. Options give the holder the right (but not the obligation) to buy (call) or sell (put) an underlying asset, like a stock, at a specified price (often refeered to as strike price) on or before a specific date.

2.2.1 Common options

Common call and put options (with their payoff structures)[10] [8]:

Standard Options

Standard options, also known as vanilla options, are the most commonly traded derivatives in financial markets. They follow a simple and well-defined payoff structure, making them the foundation of many more complex option types. The two primary categories of standard options are European options and American options.

1. **European Options**
   - Put: Gives the holder the right to sell an asset at a fixed price K only at expiration.
   - Call: Gives the holder the right to buy an asset at a fixed price K only at expiration.
   - Payoff: $f(S_T) = \max(S_T - K, 0)$ for call and $f(S_T) = \max(K - S_T, 0)$ for put.
2. **American Options**
   - Put: Gives the holder the right to sell an asset at a fixed price K at any time before or at expiration.
   - Call: Gives the holder the right to buy an asset at a fixed price K at any time before or at expiration.
   - Payoff: $f(S_T) = \max(S_T - K, 0)$ for call and $f(S_T) = \max(K - S_T, 0)$ for put.

Path-Dependent Options

These options depend on the price path of the underlying asset during the option's lifetime rather than just the final price at expiration.

1. **Asian Options**
   - Put: Gives the holder the right to sell an asset at the average price or relative to the average price.
   - Call: Gives the holder the right to buy an asset at the average price or relative to the average price.
   - Payoff: $f(A) = \max(A - K, 0)$ for call and $f(A) = \max(K - A, 0)$ for put, where A is calculated as:
$$A = \frac{1}{n} \sum_{i=1}^{n} S_{t_i}$$
2. **Barrier Options**
   - Put: Provides the right to sell an asset if the barrier condition is met.
   - Call: Provides the right to buy an asset if the barrier condition is met.
   - Payoff for put:
$$f(S_T) = \begin{cases} \max(K - S_T), & \text{if the barrier condition is met} \\ 0, & \text{otherwise} \end{cases}$$

- Payoff for call:

$$f(S_T) = \begin{cases} \max(S_T - K), & \text{if the barrier condition is met} \\ 0, & \text{otherwise} \end{cases}$$

3. **Lookback Options**
   - These options depend on the maximum or minimum price of the underlying during the option's life.
   - Call (Floating Strike): $f(S_T) = S_T - \min_{t\in[0,T]} S(t)$
   - Put (Floating Strike): $f(S_T) = \max_{t\in[0,T]} S(t) - S_T$
   - Call (Fixed Strike): $f(S_T) = \max\left(\max_{t\in[0,T]} S(t) - K, 0\right)$
   - Put (Fixed Strike): $f(S_T) = \max\left(K - \min_{t\in[0,T]} S(t), 0\right)$

Multi-Asset Options

These options depend on multiple underlying assets rather than just one.

1. **Basket Options**
   - These depend on the weighted average price of multiple underlying assets.
   - Call: $f(S_T) = \max\left(\sum_{i=1}^{n} w_i S_i(T) - K, 0\right)$
   - Put: $f(S_T) = \max\left(K - \sum_{i=1}^{n} w_i S_i(T), 0\right)$
   - Here, w_i are the weights of the assets in the basket, $S_i(T)$ is the price of the i-th asset at maturity, and K is the strike price.
2. **Rainbow Options**
   - Rainbow options involve multiple underlying assets and often focus on the best or worst-performing asset.
   - Call (Best Asset): $f(S_T) = \max\left(\max_{i=1,\ldots,n} S_i(T) - K, 0\right)$
   - Put (Best Asset): $f(S_T) = \max\left(K - \max_{i=1,\ldots,n} S_i(T), 0\right)$
   - Call (Worst Asset): $f(S_T) = \max\left(\min_{i=1,\ldots,n} S_i(T) - K, 0\right)$
   - Put (Worst Asset): $f(S_T) = \max\left(K - \min_{i=1,\ldots,n} S_i(T), 0\right)$

2.2.2 Black Scholes Model and Analytical Pricing

The Black Scholes model, introduced in 1973 by Fischer Black and Myron Scholes, provides a good analytical tool for pricing European options. It assumes that the underlying asset follows a GBM with constant volatility, drift, and no arbitrage opportunities. The price of a European call option is given by:

$$C = S_0\Phi(d_1) - Ke^{-rT}\Phi(d_2),$$

where:

$$d_1 = \frac{\ln(S_0/K) + (r + \sigma^2/2)T}{\sigma\sqrt{T}}, \quad d_2 = d_1 - \sigma\sqrt{T},$$

and Φ represents the cumulative distribution function of the standard normal distribution. The price of a European put option can be derived using put-call parity.

However, the assumptions of the Black Scholes model make it unsuitable for more complex options like Asian, barrier, or lookback options. These limitations lead to the need for numerical methods. [10][8]

2.2.3 Exact Model for Barrier Option

We consider the Black Scholes model where the underlying asset follows a GBM. The closed-form solution for an up-and-out barrier call option with spot price S, strike K, barrier B, maturity T, risk-free rate r, and volatility σ is given by:

$$\text{UO\_Barrier}(S, T, K, r, \sigma, B) = S(A_1 + A_2) - e^{-rT}K(A_3 + A_4)$$

where:

$$\begin{aligned}
d_{\pm}(z) &= \frac{\ln z + \left(r \pm \frac{1}{2}\sigma^2\right) T}{\sigma\sqrt{T}} \\
A_1 &= \Phi\left(d_+\left(\frac{S}{K}\right)\right) - \Phi\left(d_+\left(\frac{S}{B}\right)\right) \\
A_2 &= -\left(\frac{B}{S}\right)^{1+\frac{2r}{v^2}} \left[\Phi\left(d_+\left(\frac{B^2}{KS}\right)\right) - \Phi\left(d_+\left(\frac{B}{S}\right)\right)\right] \\
A_3 &= \Phi\left(d_-\left(\frac{S}{K}\right)\right) - \Phi\left(d_-\left(\frac{S}{B}\right)\right) \\
A_4 &= -\left(\frac{S}{B}\right)^{1-\frac{2r}{v^2}} \left[\Phi\left(d_-\left(\frac{B^2}{KS}\right)\right) - \Phi\left(d_-\left(\frac{B}{S}\right)\right)\right]
\end{aligned}$$

Here, $\Phi(\cdot)$ denotes the cumulative distribution function of the standard normal distribution. This formula will be used to compare the estimated price!

2.2.4 Other Common Pricing Models

Stochastic Volatility Models

Stochastic Volatility Models extend the Black–Scholes framework by modeling volatility as a separate random process. This approach captures observed market phenomena such as volatility clustering and the implied volatility smile, at the cost of increased computational complexity [25].

- **Description:** The underlying asset price follows an SDE with an additional SDE controlling volatility.
- **Pros:** More realistic, can fit market option prices better.
- **Cons:** Additional state variables => more complex to simulate.

2.2.5 Numerical Techniques

Binomial Tree Method

A popular numerical approach for pricing options is the binomial (or lattice-based) tree method. The idea is to discretize time into small steps, and at each step, the underlying

price can move "up" or "down" with specified probabilities. Figure 2.2 shows a simple example with three time steps.

- **Use Case:** Particularly useful for American or path-dependent options that require early-exercise checks at discrete times.
- **Limitations:** Can become computationally large in high dimensions or with many time steps.

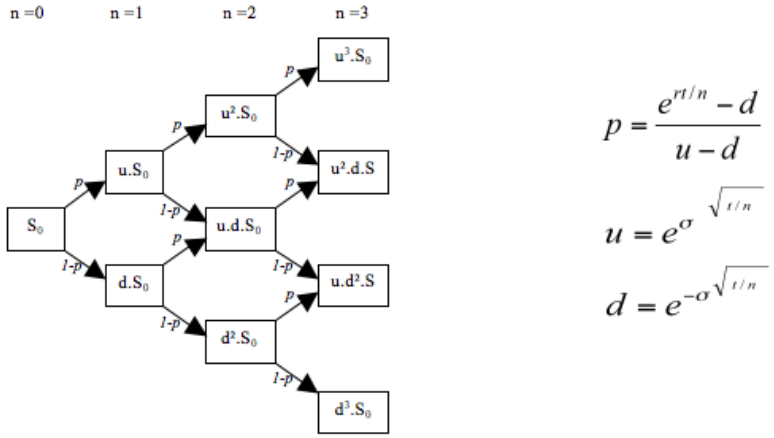


Figure 2.2: Visualization of a binomial tree for 3 time steps. (Source: Wikipedia)

Finite Difference Methods

Finite difference methods are another numerical approach for solving the underlying partial differential equation(s) (such as the Black–Scholes PDE). They discretize both time and space (the price dimension) into a grid. [7]

- **Algorithmic Idea:** Approximate derivatives $\frac{\partial^2}{\partial S^2}$, $\frac{\partial}{\partial S}$, and $\frac{\partial}{\partial t}$ via difference quotients.
- **Pros:** Structured PDE-based approach, often quite accurate for lower dimensional (1D or 2D) problems.
- **Cons:** The computational grid grows quickly with dimension, making it impractical for many-underlying (high-dimensional) options.

2.2.6 Monte Carlo Simulation for Option Pricing

Monte Carlo simulation is a powerful numerical method that estimates the value of options by simulating a large number of possible paths for the underlying asset. Unlike analytical solutions, MC methods are highly flexible and can accommodate path-dependent options, multiple underlying assets, and stochastic volatility.

Monte Carlo methods are particularly useful for:

- Path-Dependent Options: Options like Asian or lookback options depend on the history of the underlying asset's price, which is easily captured through simulation.

- Multi-Dimensional Problems: Basket and rainbow options involve multiple underlying assets, making analytical solutions infeasible.
- Flexibility: The method is highly adaptable to different models, including stochastic volatility and jump-diffusion models.

2.2.7 Pricing Techniques for Specific Options

For the different types of options, the following approaches are commonly used:

- European Options: For plain European options under the Black–Scholes assumptions, the Black–Scholes formula provides an analytic price. MC will converge to the same result but is typically unnecessary when a closed form solution exists[3] [11].
- American Options: An American option can be exercised at any time before expiry, which invalidates the closed form Black–Scholes formula. Thus, pricing American options usually relies on numerical methods. MC methods or lattice approaches (e.g. a binomial tree) are commonly used to handle the early exercise feature[6] [21].
- Barrier Options: Barrier options pay off depending on whether the underlying asset price crosses a predetermined barrier level during the option's life. MC simulation can price barrier options by tracking the underlying path and checking the barrier condition at each time step. (Finite-difference methods or analytical formulas exist for some simple barrier contracts, but in complex cases simulation is needed)[22][8]
- Asian Options: Asian options depend on the average price of the underlying asset over a period of time. MC simulation naturally accommodates this by averaging the prices along each simulated path and then computing the payoff accordingly[16].
- Lookback Options: Lookback options have payoffs based on the maximum or minimum asset price attained during the option's life. MC can capture this by recording the extremal values along each path and evaluating the payoff from those extrema[9].
- Basket and Rainbow Options: These options involve multiple underlying assets rather than just one. For example, a basket option's payoff might depend on a weighted average of several asset prices, whereas a rainbow option's payoff could depend on the best or worst performer among the assets. In general, closed form solutions do not exist for multi asset options, so MC simulation is often the method of choice for pricing them. It can readily handle the high-dimensional integration required for basket and rainbow payoffs, albeit with increased variance[32][4].

2.2.8 Advantages/Disadvantages of Monte Carlo over Other Pricing Models

While the Black Scholes model is central in financial engineering, it falls short for options with features like path dependency or multiple underlying assets. Monte Carlo methods overcome these limitations by directly simulating the dynamics of the underlying assets, offering:

- Realistic Modeling: Ability to incorporate more sophisticated models for asset dynamics, such as stochastic volatility or jump processes.
- Broad Applicability: Effective for a wide range of options, including exotic options that cannot be priced analytically.

- Accuracy: Low accuracy with sufficient computational resources and variance reduction techniques like antithetic variates and control variates.

Monte Carlo simulation provides a robust and flexible framework for option pricing, making it a key tool for complex financial instruments in modern risk management and financial engineering. This is the reason why Monte Carlo methods will be used in this project.

2.3 Challenges with Monte Carlo

The standard MC have certain challenges when it comes to computational cost and accuracy. Thus, MC is usually only used for problems in high dimension when other methods are worse.

2.3.1 Computational cost

The convergence rate of standard MC methods is primarily derived from the Law of Large Numbers and the Central Limit Theorem. Given an expectation of the form:

$$I = \mathbb{E}[f(X)] = \int f(x)p(x)dx, \tag{2.14}$$

where X is a random variable drawn from a probability density function $p(x)$, the MC estimator is given by:

$$I_N = \frac{1}{N}\sum_{i=1}^{N} f(X_i), \tag{2.15}$$

where $X_1, X_2, \ldots, X_N$ are independent and identically distributed (i.i.d.) samples from $p(x)$. By the Strong Law of Large Numbers (SLLN), it follows that:

$$\lim_{N\to\infty} I_N = I, \quad \text{a.s} \tag{2.16}$$

To analyze the convergence rate, the error is defined as:

$$\epsilon_N = I - I_N. \tag{2.17}$$

Applying the Central Limit Theorem, the asymptotic distribution of the error is obtained:

$$\sqrt{N}(I_N - I) \xrightarrow{d} \mathcal{N}(0, \sigma^2), \tag{2.18}$$

where $\sigma^2 = \text{Var}(f(X))$. This result indicates that the standard deviation of the error is:

$$\mathbb{E}[|I_N - I|^2]^{1/2} = O(N^{-1/2}). \tag{2.19}$$

Thus, the convergence rate of standard MC methods is $O(N^{-1/2})$. This implies that to reduce the error by a factor of 10, one must increase the number of samples by a factor of 100 [19].

2.3.2 Accuracy

In general, it is inevitable that a MC simulation will be incorrect compared to the true value, the question is how wrong will the estimation be. There is a lot of different ways to measure accuracy, and this subsection will discuss Mean Square Error (MSE), Confidence Intervals and Variance Estimation.

Mean Square Error

The accuracy of a MC estimator is commonly measured using the Mean Squared Error (MSE), defined as:

$$\text{MSE}(I_N) = \mathbb{E}[(I_N - I)^2] = \text{Var}(I_N) + (\mathbb{E}[I_N] - I)^2. \tag{2.20}$$

Since the MC estimator is unbiased (i.e., $\mathbb{E}[I_N] = I$), the bias term vanishes, leaving:

$$\text{MSE}(I_N) = \frac{\sigma^2}{N}, \tag{2.21}$$

where σ^2 is the variance of the underlying function values. This implies that the error decreases at a rate of $O(N^{-1/2})$ as established above.

Confidence Intervals

A key feature of MC methods is the ability to construct confidence intervals for estimated quantities. By the CLT, the distribution of the MC estimator approximates a normal distribution for sufficiently large N:

$$\sqrt{N}(I_N - I) \xrightarrow{d} \mathcal{N}(0, \sigma^2). \tag{2.22}$$

This allows us to construct a confidence interval for I at a confidence level of $1 - \alpha$:

$$I_N \pm z_{\alpha/2} \frac{\sigma}{\sqrt{N}}, \tag{2.23}$$

where $z_{\alpha/2}$ is the standard normal quantile corresponding to the chosen confidence level (e.g., 1.96 for a 95% confidence interval).

Variance Estimation

To use confidence intervals in practice, the variance σ^2 must be estimated. A common estimator for the variance is the sample variance, given by:

$$\hat{\sigma}_N^2 = \frac{1}{N-1} \sum_{i=1}^{N} (f(X_i) - I_N)^2. \tag{2.24}$$

Using this estimate, the confidence interval can be rewritten as:

$$I_N \pm 1.96 \frac{\hat{\sigma}_N}{\sqrt{N}}. \tag{2.25}$$

This provides a probabilistic estimate of the error, making it a crucial tool in practical applications[19].

2.4 Generating Random Numbers

Random number generation is a fundamental component of computational methods used in finance, physics, and engineering. One of the most important aspects of MC methods are the random numbers that are generated which drives the simulations. The quality of a MC simulation is highly dependent on the properties of the underlying random number generator[8].

2.4.1 General Setting

A pseudo-random number generator (PRNG) is a deterministic algorithm that produces sequences approximating truly random numbers. The simplest PRNGs use a linear congruential generator (LCG), which follows the recurrence relation:

$$U_n = (aU_{n-1} + b) \mod m \tag{2.26}$$

where a, b, m, and an initial seed U_0 define the sequence. These numbers are uniformly distributed in $[0, 1]$ and can be transformed to follow other distributions as needed [5].

2.4.2 Skipping Ahead in PRNGs

One challenge with PRNGs is efficiently jumping ahead in the sequence, which is important for parallel computations. Instead of iterating through all steps, matrix exponentiation techniques allow direct computation of the state at step n:

$$U_n = (a^n U_0 + c) \mod m \tag{2.27}$$

where c is precomputed based on modular arithmetic. This enables efficient division of PRNG sequences among multiple computing nodes which is crucial for large scale MC simulations [5].

2.4.3 Inverse Transform Method

The Inverse Transform Method is a fundamental approach for generating random variables from non-uniform distributions. It relies on the fact that if $U \sim U(0, 1)$, then the random variable:

$$X = F^{-1}(U) \tag{2.28}$$

follows the desired distribution $F(x)$, provided $F(x)$ is continuous and invertible. This method is particularly useful for generating exponential and normal distributions [5].

2.4.4 Acceptance-Rejection Method

The Acceptance-Rejection Method is an alternative when the inverse CDF is difficult to compute. It involves:

1. Sampling Y from an easy-to-sample proposal distribution $g(y)$.
2. Computing an acceptance probability: $p = f(Y)/(Mg(Y))$, where $f(y)$ is the target density function and M is a bounding constant.
3. Accepting Y with probability p, otherwise repeating the process.

This method is often used for distributions with complex shapes that are difficult to sample from directly [5].

2.4.5 Generating Multivariate Normals

Generating multivariate normal random vectors $X \sim \mathcal{N}(\mu, \Sigma)$ is done by:

1. Sampling a standard normal vector $Z \sim \mathcal{N}(0, I)$.
2. Transforming it using the Cholesky decomposition of the covariance matrix $\Sigma = LL^T$, so that:

$$X = LZ + \mu \tag{2.29}$$

This ensures X has the desired mean μ and covariance Σ. This technique is widely used in financial risk modeling and portfolio simulations [5].

2.5 Error Reduction Techniques

As established in section 2.3, there exists some issues with standard MC simulations in terms of computational cost and accuracy. One solution to this problem is to use Variance Reduction Techniques (VRT)[27].

2.5.1 Quasi Monte Carlo (QMC)

Quasi Monte Carlo is, compared to the standard Monte Carlo method, a deterministic alternative. It uses low-discrepancy sequences instead of random sampling which fills the sample space more uniformly and thus reduces the variance. [12]

For QMC the convergence rate is generally $O(N^{-1})$. Although there are ways to achieve better convergence rate. For functions of bounded variation the theory shows that a convergence rate of $O(N^{-1}log^s N)$, where s is the dimension, is possible. However, it is possible to improve the convergence rate even further by introducing additional smoothness parameters or techniques like scrambling, higher-order digital nets and lattice rules. Thus, the convergence rate (if using some of the mentioned parameters/techniques) is $O(N^{-\frac{3}{2}}log^s N)$ and can get as high as $O(N^{-2})$. Lastly, if Randomized Quasi Monte Carlo (RQMC) is used, the convergence rate can get as high as $O(N^{-\frac{3}{2}})$ [18].

2.5.2 Low-Discrepancy Sequences

Low-discrepancy sequences (LDS) play a crucial role in QMC methods because it offers a deterministic approach to numerical integration and simulation. Unlike pseudo random numbers which may exhibit clustering and irregularities low-discrepancy sequences are designed to be well distributed over the unit hypercube. The effectiveness of QMC methods in reducing variance is largely determined by the properties of these sequences.

Let $U_1, \ldots, U_N$ be i.i.d. vectors, each uniformly distributed in the D-dimensional unit hyper-cube. Then the MC estimator for the integral:

$$V_0 \;=\; \int_{[0,1)^D} h(u)\, du.$$

becomes:

$$\widehat{V}_0 \;=\; \frac{1}{N}\sum_{i=1}^{N} h(U_i).$$

The discrepancy is defined as:

$$D\big((u_i)_{1\le i\le N}\big) \;=\; \sup_{E=[0,t_1)\times\cdots\times[0,t_D)} \left| \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}_{\{u_i\in E\}} \;-\; \mathrm{VOL}(E) \right|, \qquad \mathrm{VOL}(E) = \prod_{k=1}^{D} t_k.$$

The famous Koksma–Hlawka inequality provides a deterministic error bound:

$$\big|\bar{V}_0 - V_0\big| \;\le\; V(h)\, D\big(\{U_i\}_{i\le N}\big),$$

where

- $V(h)$ is the *variation* of the integrand h (hard to alter), and
- $D(\cdot)$ is the *discrepancy* of the point set, i.e. a quantitative measure of how evenly the points cover the hyper cube.

A fundamental justification for the use of low-discrepancy sequences is provided by the Koksma-Hlawka inequality, which establishes a bound on the integration error based on the discrepancy of the sequence and the variation of the integrand. The discrepancy of a sequence quantifies how evenly it covers the domain, with lower values indicating better distribution properties. For a given s, the best achievable order of the discrepancy is $O(N^{-1}log^{s}N)$, a characteristic feature of low-discrepancy sequences[30].

There exist many families of low-discrepancy sequences, including Halton, Faure, Niederreiter, and Sobol' sequences, each with distinct construction methods and efficiency in high-dimensional settings. Among these, Sobol' sequences is the most widely used due to their adaptability, superior numerical properties, and effectiveness in high-dimensional applications [1].

Sobol'

Sobol' sequences are a widely used low-discrepancy sequence in QMC methods and one of the most common low-discrepancy within the finance sector.

A key advantage of Sobol' sequences in option pricing is their ability to improve convergence rates compared to standard MC sampling. By using deterministic sequences with low discrepancy properties, they achieve faster error reduction, making them effective for pricing high dimensional derivatives. This efficiency is particularly relevant when estimating Greeks, where variance reduction is crucial for accurate sensitivity analysis.

Compared with pseudo-random sampling the variance of a QMC estimator built on Sobol' points drops at close to $\mathcal{O}(N^{-1})$ rather than $\mathcal{O}(N^{-1/2})$. In option pricing, where hundreds or thousands of Gaussian variates may be required per path, that faster convergence translates into huge CPU savings.

A single Sobol' draw delivers a full vector of uniform variates $\mathbf{u}^{(i)} = (u_1^{(i)}, \ldots, u_d^{(i)}) \in [0,1]^d$, whose length d is chosen to equal the number of random numbers required for one simulation path (typically the number of time steps or factors in the model). In practice the Sobol' generator is most uniform after every base-2 extension, so practitioners run with $N = 2^k$ points and usually drop the very first point (the all zero vector). The effective path count is therefore $N_{\text{paths}} = 2^k - 1$, a choice that (i) avoids a degenerate zero shock on the first path and (ii) guarantees the lowest possible discrepancy for every prefix of the sequence, leading to the tightest error bands in QMC tests [30].

Sobol' is not one monolithic list of D-vectors. It is D separate sequences $(x_i^{(d)})_{i\geq 0}$ with each $x_i^{(d)} \in (0,1)$. Taking the i-th element from every coordinate produces the i-th D-vector. If an application later needs a higher dimension $D' > D$, the first D coordinates remain unchanged which means no regeneration required.

Each coordinate maintains an unsigned 32-bit integer state $y_i^{(d)} \in \{0, \ldots, 2^{32} - 1\}$ and scales it to the unit interval via:

$$x_i^{(d)} = \frac{y_i^{(d)}}{2^{32}} \in (0,1). \tag{2.30}$$

For every axis d pre-compute 32 direction numbers $DN_k^{(d)}$, $k = 0, \ldots, 31$. Let J_i be the index of the right-most zero bit in the binary representation of i. Starting from $y_0^{(d)} = 0$ the update rule is

$$y_{i+1}^{(d)} \;=\; y_i^{(d)} \;\oplus\; DN_{J_i}^{(d)}, \tag{2.31}$$

where $\oplus$ denotes bit-wise exclusive-or. In practice, the pair $(y_0^{(d)}, x_0^{(d)}) = (0, 0)$ is skipped, so simulation begins with index $i = 1$. Because XOR is associative and a single CPU instruction, the cost of producing the next point is negligible.

Direction number $DN_k^{(d)}$ flicks in and out of the state every 2^{k+1} points (DN_0 every two points, DN_1 every four points, DN_2 every eight points, and so on). Thus the first 2^k points realise all 2^k combinations of the first k direction numbers exactly once. That property explains the sequence's extremely low discrepancy. A schematic of the first sixteen integers on one axis is often shown to illustrate the pattern [30]:

i	DN0	DN1	DN2	DN3	resulting y
0	0	0	0	0	0
1	1	0	0	0	DN0
2	1	1	0	0	DN0+DN1
3	0	1	0	0	DN1
4	0	1	1	0	DN1+DN2
5	1	1	1	0	DN0+DN1+DN2
6	1	0	1	0	DN0+DN2
7	0	0	1	0	DN2
8	0	0	1	1	DN2+DN3
...					

Additionally, techniques like scrambling can be applied to Sobol' sequences to further improve their performance. Scrambling introduces controlled randomness, lowering the risk of bias while preserving the low discrepancy properties, which is beneficial when dealing with discontinuous payoff functions common in financial derivatives [1].

2.5.3 Bias-Variance Tradeoff

When applying MC methods for option pricing, particularly with work reduction or approximate modeling, there is a risk of introducing bias to reduce computation time. According to [31], such techniques can lower the cost per simulated path (and thus allow more paths under a strict computational budget), but may introduce small systematic errors. In these cases, the total estimation error is better measured via the mean squared error (MSE), which accounts for both the variance $\mathrm{Var}(\hat{\theta})$ and the squared bias $(\mathrm{Bias}(\hat{\theta}))^2$:

$$\mathrm{MSE}(\hat{\theta}) \;=\; \mathrm{Var}(\hat{\theta}) \;+\; \big(\mathrm{Bias}(\hat{\theta})\big)^2. \tag{2.32}$$

[31] also points out:

- Standard MC estimators for option pricing are usually unbiased when the underlying stochastic model is discretized finely enough, making variance the main source of error.
- Approximate models (such as fewer time steps, simplified factor structures, or cheaper function evaluations) can introduce a small bias. However, the overall MSE can still decrease if the variance reduction (through faster runs or larger sample sizes) outweighs the added bias.

- In high-dimensional or frequently repeated simulations (daily pricing, risk management etc), accepting slight bias for a notable reduction in variance often gives lower MSE overall.

As shown in [31], short-cutting computations such as employing fewer simulation steps, using partial factor models or adopting simplified function approximations can be justified if the resulting bias remains small enough that mean squared error still beats that of a fully accurate but high variance simulation.

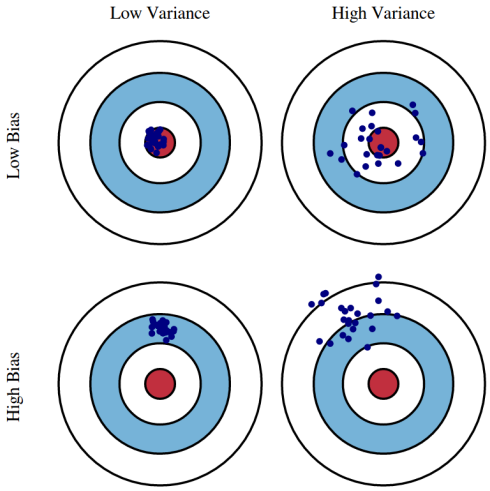


Figure 2.3: Illustration of Bias/Variance

2.6 Stochastic Volatility Models for Option Pricing

Until now, processes with a constant volatility has been discussed. However, there exists better ways to model the future price of an asset. In a stochastic volatility model, not just the evolution of the prince is random but the volatility itself varies randomly throughout the time. This allows for a more realistic model when pricing financial derivatives [8].

2.6.1 Heston Model

One of the most common stochastic volatility models is the Heston model. It is designed to overcome the constant volatility assumption of the Black Scholes framework[15].

Model Setup

Under the Heston model, the price process S_t and its variance V_t are modeled by two stochastic differential equations (SDEs):

$$dS_t = \mu\, S_t\, dt \;+\; \sqrt{V_t}\, S_t\, dW_t^{(S)}, \tag{2.33}$$

$$dV_t = \kappa(\theta - V_t)\, dt \;+\; \sigma_V \sqrt{V_t}\, dW_t^{(V)}, \tag{2.34}$$

and their correlation is:

$$dW_t^{(S)}\, dW_t^{(V)} = \rho\, dt. \tag{2.35}$$

[15] also states the following results regarding the Heston Model:

- The Heston model allows volatility to be both random and mean-reverting which is capturing key empirical phenomena (implied volatility smiles/skews).
- Calibration of the model typically reveals that the correlation ρ is often negative, reflecting the "leverage effect" (volatility tends to increase after price drops).
- Compared to the Black Scholes model, the Heston model is better at fitting market option prices across strikes and maturities, but it has more parameters and computational overhead.

2.6.2 Constant Elasticity of Variance Models (CEVM)

The Constant Elasticity of Variance model is a one-dimensional diffusion process for an asset price S_t, where the instantaneous volatility is a power function of the price:

$$\sigma(S) \;=\; a\, S^{\beta}, \tag{2.36}$$

with $a > 0$ as the volatility scale parameter and β the elasticity parameter. Under this specification, the asset price S_t follows the SDE

$$dS_t \;=\; \mu\, S_t\, dt \;+\; a\, S_t^{\beta+1}\, dB_t, \tag{2.37}$$

where B_t is a standard Brownian motion and μ is the drift term (which may be taken under the risk-neutral or real-world measure, depending on the application).

2.7 Brownian Bridge Construction

A convenient and instructive way to generate a Brownian Motion path on $[0, T]$ is via the Brownian Bridge (BB) Construction. The key aspects of this method is to build the path level by level, progressively "filling in" midpoints and ensuring that, at each stage, the joint distribution of the constructed process at sampled times matches that of standard Brownian Motion[2].

Intuitively, the bridge samples the most influential points first: starting with the two end-points W_0 and W_T; next the mid-point $W_{T/2}$ is drawn, then the quarter-points $W_{T/4}, W_{3T/4}$, and so on, halving every remaining interval at each stage. This dyadic refinement orders the variance contributions so that the early Gaussian shocks explain the bulk of the path's variability, while later shocks merely add fine detail. Feeding these successively less- important shocks into the higher Sobol' dimensions aligns the Brownian path's variance hierarchy with the discrepancy profile of the low-discrepancy sequence and is the main reason the Sobol+Bridge combination is so effective for barrier and lookback options [8].

Step 1: Setup of Dyadic Time Intervals

Fix a final time $T > 0$. For each integer $n \geq 0$, partition the interval $[0, T]$ into 2^n dyadic intervals of equal length

$$\Delta t_n \;=\; \frac{T}{2^n}.$$

Specifically, let

$$t_{n,k} \;=\; k\, \Delta t_n, \quad k = 0, 1, \ldots, 2^n,$$

so that

$$I_{n,k} \;=\; [\, t_{n,k},\; t_{n,k+1}\,] \;=\; \left[k\, \frac{T}{2^n},\; (k+1)\, \frac{T}{2^n} \right], \quad k = 0, 1, \ldots, 2^n - 1.$$

Going from level n to level $n+1$ subdivides each interval $I_{n,k}$ exactly in half:

$$I_{n,k} \;=\; I_{n+1,2k} \;\cup\; I_{n+1,2k+1},$$

where each new interval at level $n+1$ has length $\Delta t_{n+1} = T/2^{n+1}$[2].

Step 2: Iterative Construction of the Path

It is constructed, level by level, an approximate Brownian Motion $\{W_{n,t}\}$ on the dyadic points $\{t_{n,k}\}$, ensuring that the values match the correct finite dimensional distributions.
Level 0. Set

$$W_{0,0} \;=\; 0 \quad \text{and} \quad W_{0,T} \;=\; \sqrt{T}\, Z_0,$$

where $Z_0 \sim \mathcal{N}(0,1)$.

Filling midpoints. For each interval $I_{n,k}$ at level n, with endpoints $t_{n,k}$ and $t_{n,k+1}$, $W_{n,\,t_{n,k}}$ and $W_{n,\,t_{n,k+1}}$ are already know. Define the midpoint

$$m_{n,k} \;=\; \frac{1}{2}\Big(t_{n,k} + t_{n,k+1}\Big) \;=\; t_{n,k} + \Delta t_{n+1}.$$

It is sampled

$$W_{n+1,\,m_{n,k}} \;=\; \frac{W_{n,\,t_{n,k}} + W_{n,\,t_{n,k+1}}}{2} \;+\; \sqrt{\frac{\Delta t_{n+1}}{2}}\, Z_{n+1,\,2k+1},$$

where $\{Z_{n+1,\,2k+1}\}$ are i.i.d. standard normal variables, independent of all previously used random variables.

Brownian Bridge Conditional Formula

Given two time points $t_1 < t_2$ and the values of Brownian motion at those points, $W(t_1) = w_1$ and $W(t_2) = w_2$, the Brownian Bridge provides the distribution of $W(t_m)$ at the midpoint $t_m = \frac{t_1+t_2}{2}$.

By the properties of Brownian Motion, the conditional distribution of $W(t_m)$ given its values at the endpoints is Gaussian with:

- **Mean** (linear interpolation):

$$\mathbb{E}[W(t_m) \mid W(t_1) = w_1,\, W(t_2) = w_2] = \frac{t_2 - t_m}{t_2 - t_1} w_1 + \frac{t_m - t_1}{t_2 - t_1} w_2$$

- **Variance** (quadratic interpolation):

$$\mathrm{Var}[W(t_m) \mid W(t_1),\, W(t_2)] = \frac{(t_m - t_1)(t_2 - t_m)}{t_2 - t_1}$$

Sampling from this conditional distribution is implemented as:

$$W(t_m) = \frac{w_1 + w_2}{2} + \sqrt{\frac{(t_m - t_1)(t_2 - t_m)}{t_2 - t_1}}\, Z, \quad Z \sim \mathcal{N}(0,1).$$

This expression is used recursively in the Brownian Bridge construction to sample midpoints of each interval, refining the path while maintaining consistency with the distribution of Brownian Motion.

Next level. Once the midpoint of every interval at level n is set, we have specified $\{W_{n+1,\,t_{n+1,k}}\}$ on a finer grid of size $2^{n+1}+1$. This procedure is repeated to advance from level $n+1$ to level $n+2$, thus obtaining increasingly refined piecewise linear paths that converge to a Brownian Motion in the limit [2].

Barrier crossing between monitoring dates

Consider an up–out barrier at level B (in log–space $b = \ln B$). Let $X_t = \ln S_t$ be the log-price, and suppose that at two consecutive monitoring times $t_i < t_{i+1}$. We have $X_i := X_{t_i} < b$ and $X_{i+1} := X_{t_{i+1}} < b$. Conditionally on these end-points, $\{X_t\}_{t_i \le t \le t_{i+1}}$ is a Brownian bridge with variance parameter $\sigma^2 \Delta t$, $\Delta t = t_{i+1} - t_i$. By the reflection principle the exact probability that this bridge ever reaches the barrier in the open interval (t_i, t_{i+1}) is

$$p_{\text{hit}} \;=\; \exp\Big(-\,\frac{2\,(b - X_i)\,(b - X_{i+1})}{\sigma^2\,\Delta t}\Big).$$

Simulation rule. At every step one extra uniform variable $U \sim \mathcal{U}(0,1)$ is drawn. If both end-points stay below the barrier the path is declared knocked out whenever $U < p_{\text{hit}}$. For numerical stability the implementation compares $\ln U$ with the log-probability $-\frac{2\,(b-X_i)\,(b-X_{i+1})}{\sigma^2\,\Delta t}$, avoiding underflow when the exponent is large in magnitude.

This additional Bernoulli test is the only place where an extra random number is consumed relative to a plain Euler discretisation and is crucial for obtaining an unbiased estimate of the barrier option price.

2.8 Discretization Methods

Monte Carlo simulations rely on numerical approximations of SDEs to model the evolution of asset prices over time. Since exact solutions to these equations are often unavailable, discretization methods become essential for practical implementations.

2.8.1 Euler-Maruyama method

The Euler-Maruyama method is a fundamental numerical method for simulating solutions of SDEs. Consider an SDE of the form:

$$\mathrm{d}X(t) = f\big(t, X(t)\big)\,\mathrm{d}t + g\big(t, X(t)\big)\,\mathrm{d}W(t), \tag{2.38}$$

where $W(t)$ is a standard Wiener process. By discretizing time into steps of size τ, with Wiener increments approximated as $\Delta W_n \sim \sqrt{\tau}\,\mathcal{N}(0,1)$, the Euler-Maruyama method iterates according to

$$X_{n+1} = X_n + \tau f\big(t_n, X_n\big) + g\big(t_n, X_n\big)\,\Delta W_n. \tag{2.39}$$

This method is the stochastic analog of the standard Euler method used for ordinary differential equations which is obtained by approximating the deterministic integral via the left-point rule and using a similar approximation for the stochastic integral.

Theorem (Strong Convergence of Euler-Maruyama). Suppose the drift f and diffusion g in the SDE

$$\mathrm{d}X(t) \;=\; f\big(t, X(t)\big)\,\mathrm{d}t \;+\; g\big(t, X(t)\big)\,\mathrm{d}W(t)$$

are globally Lipschitz, among other regularity conditions. Then the Euler-Maruyama approximation X_n converges to the true solution X with strong order $\frac{1}{2}$. Concretely, there is a constant C (independent of the step size Δt) such that

$$\left(\mathbb{E}\,\|X(t_n) - X_n\|^2\right)^{\frac{1}{2}} \;\leq\; C\,(\Delta t)^{\frac{1}{2}},$$

meaning the pathwise (strong) error scales like $\sqrt{\Delta t}$ [17].

Despite its simplicity, Euler-Maruyama has only a strong order of convergence of $\frac{1}{2}$, meaning that the pathwise error scales as $O(\sqrt{\tau})$.

Theorem (Weak Order of Convergence). A discretization scheme is said to have weak order of convergence $\beta > 0$ if there exists a constant C such that for all sufficiently smooth test functions f,

$$|\mathbb{E}\left[f(X_n)\right] - \mathbb{E}\left[f(X(t_n))\right]| \leq C(\Delta t)^{\beta}$$

for all sufficiently small step sizes Δt [8]. Here, the set of test functions f typically consists of functions whose derivatives up to order $2\beta + 2$ are polynomially bounded. However, its weak order of convergence is often 1, making it particularly useful for applications where expected values, such as option prices in financial models, are of interest.

In mathematical finance, the Euler-Maruyama method is commonly applied to simulate asset price variations under models such as Black Scholes and Heston. For instance, applying the Euler discretization to the Black Scholes SDE:

$$\mathrm{d}S_t = rS_t\mathrm{d}t + \sigma S_t\mathrm{d}W_t, \tag{2.40}$$

yields the discrete update rule:

$$S_{t+\tau} = S_t + rS_t\tau + \sigma S_t\sqrt{\tau}Z, \tag{2.41}$$

where $Z \sim \mathcal{N}(0,1)$. Similarly, for stochastic volatility models like Heston's, additional modifications are required to prevent negative variance values, such as full truncation or reflection schemes.

While the Euler-Maruyama method is widely used due to its computational efficiency, it has some limitations. For instance, it tends to introduce bias in the simulation of processes with multiplicative noise or nonlinearity, thus requiring careful step size selection to maintain accuracy. Furthermore, in models with significant stochasticity, higher-order schemes like the Milstein method can offer improved convergence without substantial increases in computational cost.

Despite these drawbacks, the Euler-Maruyama method remains a go-to approach for simulating stochastic processes, especially in finance and other applied fields where approximate solutions suffice. [14][29].

2.8.2 Milstein Scheme

The Milstein scheme is a discretization technique for stochastic differential equations (SDEs) that improves upon the simpler Euler scheme by adding a correction term derived from Ito's lemma. It is particularly useful for SDEs whose diffusion coefficient depends on the state variable, as it helps reduce the local discretization error compared to Euler.

Consider an SDE of the form:

$$dS_t = a(S_t)dt + b(S_t)dW_t, \tag{2.42}$$

where $a(S_t)$ and $b(S_t)$ may depend on S_t (but not explicitly on t), and W_t is a Brownian motion. The integral form is:

$$S_{t+\Delta t} = S_t + \int_t^{t+\Delta t} a(S_s)ds + \int_t^{t+\Delta t} b(S_s)dW_s. \tag{2.43}$$

The Milstein scheme improves compared to Euler by expanding $b(S_t)$ via Ito's lemma and keeping a mixed Brownian term (i.e., the $dW_t dW_t$ term). This generates an additional correction factor that accounts for how $b(S_t)$ itself changes with S_t. To derive the expression, first apply an Ito expansion to $b(S_t)$. This effectively gives:

$$db(S_t) = \left(b'(S_t)a(S_t) + \frac{1}{2}b''(S_t)(b(S_t))^2\right)dt + b'(S_t)b(S_t)dW_t, \tag{2.44}$$

where b' and b'' are the first and second derivatives of b with respect to S. The Milstein discretization for a single time step Δt then becomes:

$$S_{t+\Delta t} = S_t + a(S_t)\Delta t + b(S_t)\sqrt{\Delta t}Z + \frac{1}{2}b(S_t)b'(S_t)\Delta t(Z^2 - 1), \tag{2.45}$$

where $Z \sim N(0,1)$. The extra term $\frac{1}{2}b(S_t)b'(S_t)\Delta t(Z^2-1)$ is the correction that distinguishes Milstein's method from Euler's.

Euler Scheme:

$$S_{t+\Delta t} = S_t + a(S_t)\Delta t + b(S_t)\sqrt{\Delta t}Z. \tag{2.46}$$

Milstein Scheme:

$$S_{t+\Delta t} = S_t + a(S_t)\Delta t + b(S_t)\sqrt{\Delta t}Z + \frac{1}{2}b(S_t)b'(S_t)\Delta t(Z^2 - 1). \tag{2.47}$$

While the Euler scheme is simpler, the Milstein scheme has higher strong-order accuracy when $b(S_t)$ is not constant (i.e., truly depends on S_t)[29].

2.8.3 Long Jump

In the context of simulating asset price paths under the risk-neutral measure, the Geometric Brownian Motion (GBM) model assumes that the logarithm of the asset price evolves as a Brownian Motion with drift. Specifically, for time steps $t_i < t_{i+1}$, the conditional distribution of the log-price $\log S(t_{i+1})$ given $\log S(t_i)$ is normally distributed:

$$\log S(t_{i+1}) = \log S(t_i) + \left(r - \frac{1}{2}\sigma^2\right)(t_{i+1} - t_i) + \sigma\sqrt{t_{i+1} - t_i}\cdot Z_i, \qquad Z_i \sim \mathcal{N}(0,1),$$

which corresponds to formula (3.22) in [8].

This formula provides an exact simulation scheme for GBM by using the closed form solution of the SDE. Unlike approximate methods such as Euler–Maruyama, it introduces no discretization error. Since GBM has independent, normally distributed log increments, the process can be simulated exactly at any discrete time grid.

This exact scheme is particularly important for simulating path-dependent options such as barrier, where precise modeling of the path's evolution is critical to detecting barrier crossings or extrema.

In this report's implementation, the formula is used to evolve the log-price $\log S(t)$ at each time step in both standard MC and QMC simulations, with or without BB construction. It corresponds to the following SDE under the risk-neutral measure:

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

whose solution is:

$$S(t) = S(0)\exp\left(\left(r - \frac{1}{2}\sigma^2\right)t + \sigma W(t)\right).$$

3. Methodology

This chapter describes the empirical and analytical procedures adopted to answer the thesis research questions. Firstly, a preliminary literature study conducted before the research questions were finalized, was done. Secondly, the developer environment was introduced and required some time to learn. Lastly, when the environment was mastered, the study was conducted and the results were analyzed.

3.1 Previous Research

The study began with the published book [8] which, according to my supervisor, is "the bible of Monte Carlo in Financial Engineering". The theoretical foundations and practical considerations of MC methods in finance are comprehensively addressed in Glasserman's seminal work. The text introduces both the mathematical foundation of MC estimation and a broad set of variance reduction techniques designed to improve efficiency. These include control variates, antithetic sampling, importance sampling, and stratified sampling. For path dependent options, such as barrier options, Glasserman places particular emphasis on the utility of Brownian bridge constructions for improving sampling efficiency and reducing effective dimensionality.

One of the most important limitations of standard MC is its slow convergence rate, typically $O(N^{-1/2})$. This has led to growing interest in QMC methods, which replace pseudo random numbers with low-discrepancy sequences such as Sobol' or Halton, in order to achieve a more uniform coverage of the sample space. While QMC is not guaranteed to outperform MC in all cases, studies such as those by Paskov and Traub [24] and Papageorgiou and Traub [23] have shown that for many financial problems, especially when combined with BB or principal component techniques, QMC methods can significantly outperform standard MC in both accuracy and efficiency.

More recently, empirical research has focused on applying QMC methods to complex derivative pricing problems. For example, Wang and Caflisch [33] demonstrated that QMC combined with effective dimension reduction strategies performs well for various types of barrier and Asian options. Lemieux [20] has also provided a detailed overview of how QMC methods can be adapted and optimized for high-dimensional financial applications.

In the context of this thesis, these prior contributions serve as a foundation for the empirical investigation. The aim is not to develop new simulation techniques, but rather to benchmark and compare the performance of standard Monte Carlo, QMC, and hybrid methods (blending QMC and MC) across different parameter regimes relevant to barrier options. By systematically varying inputs such as strike, barrier level, and volatility, the study evaluates the stability, accuracy, and convergence behavior of each method. In particular, it seeks to determine the practical benefits of combining QMC with Brownian bridge in a modern pricing environment such as Quantlab.

3.2 The Development Environment: Qlang

3.2.1 What is Qlang?

Qlang is a specialized environment for quantitative finance simulations. It provides

- Built-in random number generators (e.g., `rng`, `sobol_gen`),
- Charting functionalities for plotting error curves or paths,
- Specialized data types like `vector(number)`, `point_number`, etc.

These features allow for straightforward MC and QMC implementations and easy financial calculations.

3.2.2 Why Qlang?

Qlang was provided by Algorithmica Research AB, and it has built-in library features for quick Sobol sequence generation, convenient random number streams and an interactive environment that makes both coding and plotting convenient. This integration streamlines the workflow compared to using a generic language, where one might need to import multiple external libraries for random numbers, charting, and data structures. Also the employee's at Algorithmica has great knowledge about the language and have always helped when necessary.

3.2.3 Key Language Constructs

Qlang uses the out function syntax to return data or produce global vector(point_number) arrays for plotting. We compile and run scripts in Qlang's IDE, which automatically manages chart displays when we define data in `vector(point_number)`. The snippet below is typical of how we define a function to produce the price of a general Geometric Brownian motion:

```
out string example_qlang_function(
    number rate,
    number vol,
    integer n_steps)
{
    // Basic setup
    number spot = 100;
    number dt   = 1.0 / n_steps;   // For demonstration
    rng my_rng  = rng(millisecond(now()));
    // We'll do a simple Euler loop for a Geometric Brownian "price"
    number current_price = spot;
    for(integer i = 1; i <= n_steps; i++)
    {
        number Z = my_rng.gauss(); // standard normal draw
        current_price *= exp((rate - 0.5 * vol * vol)*dt + vol*sqrt(dt)*Z);
    }
    // Print or store the final "price"
    return strcat(["Final price from example_qlang_function = ",
    string(current_price)]);
}
```

3.3 Overall Approach and Rationale

3.3.1 Focus on Monte Carlo

In this project, both standard MC and QMC is used to price Barrier Options. MC is especially appropriate for complex or path dependent options, where closed form solutions do not exist or are really complicated. Standard MC is flexible but can have high variance, so it is compared to QMC approaches.

3.3.2 Choice of Error Reduction / QMC

Specifically standard MC vs. QMC with a Brownian Bridge construction for barrier paths is tested. This bridging approach allows in between time-step barrier checks. From the code, it's clear that no other variance reduction methods like control variates or antithetics was incorporate; the only focus was on QMC's low-discrepancy properties as our main variance reducer.

3.4 Implementation Details

3.4.1 Code Organisation

All pricing logic lives in self-contained `out` functions so that a single call both computes and plots the result in Quantlab. The current project focuses exclusively on an up-and-out barrier call. The relevant functions are:

- `price_barrier_option(...)` – baseline MC
- `price_barrier_option_qmc_BB(...)` – Sobol'QMC with midpoint Brownian bridge
- `UO_Barrier(...)` – used as the exact benchmark
- `MC_BO_calculate_error(...)` and `QMC_BO_calculate_error(...)` – sweep the number of paths, store the relative/standard errors in global `vector(point_number)` series for automatic plotting
- Helper routines such as `buildBrownianPathMidpointFree(...)` (path constructor)

The same functions, but for average absolute/relative error exists as well and have the same construction. Each pricer takes market parameters $(S_0, K, r, \sigma, T, B)$, discretises the path, draws the appropriate random stream, accumulates discounted pay offs, and finally pushes both the estimate and its sampling error to the caller.

3.4.2 Key Functions

`price_barrier_option(...)` – **Standard MC**

- Euler steps on $\log S_t$ with drift $(r - \frac{1}{2}\sigma^2)\Delta t$ and diffusion $\sigma\sqrt{\Delta t}\, Z$
- Checks the upper barrier after every step and uses the in-between crossing probability to avoid missing hits between time points
- Returns the discounted mean, plus the usual MC standard error $\hat{\sigma}/\sqrt{N}$.

`price_barrier_option_qmc_BB(...)` – **QMC+BB**

- First $d = n_{\text{qsteps}}$ Sobol' coordinates become $N(0,1)$ draws; any remainder (if $n_{\text{steps}} > d$) falls back to a pseudo-random rng.
- `buildBrownianPathMidpointFree` generates the Wiener path on a 2^n grid, concentrating the largest variance into the lowest Sobol' dimensions.
- Barrier detection is identical to the MC routine, so the two estimators differ only by their sampling schemes.

Error and plotting helpers Both `MC_BO_calculate_error` and `QMC_BO_calculate_error` loop over an increasing path count, calling the corresponding pricer,computing

$$\text{rel\_error} = 100 \times \left|\hat{V} - V_{\text{exact}}\right| / V_{\text{exact}}, \qquad \text{std\_error} = \hat{\sigma}\, e^{-rT} / \sqrt{N},$$

and pushing those points to `MC_BO_Error_plot`, `QMC_BO_Error_plot` and their standard-error companions. Quantlab then renders the plots automatically when the user calls `show_MC_BO_Error_plot()` or the QMC companion.

3.4.3 Random Number Generators

- **Pseudo-random**: `rng my_rng = rng(millisecond(now()));`
- **Sobol'**: `sobol_gen s = new sobol_gen(n_qsteps, skip);` produces one low-discrepancy $[0,1]^{n_q steps}$ vector per simulation path; the optional `skip` lets us (i) discard the all-zero point and (ii) partition the sequence across parallel jobs.

Standard MC always uses the former; QMC replaces the whole normal stream with Sobol' points whenever possible.

3.4.4 Brownian Bridge Path Builder

The function `buildBrownianPathMidpointFree` constructs a Brownian path over the interval $[0, T]$, assuming a dyadic time grid with $n_{\text{steps}} = 2^n$. It fills in the midpoints level by level using Brownian bridge interpolation:

1. Set $W_0 = 0$ and $W_T = \sqrt{T}\, Z_0$;
2. For each refinement level $\ell = 0, \ldots, n-1$ halve every open interval and sample the midpoint with conditional mean/variance

$$\mathbb{E}[W_m] = \frac{t_2 - m}{t_2 - t_1} W_{t_1} + \frac{m - t_1}{t_2 - t_1} W_{t_2}, \quad \text{Var}[W_m] = \frac{(m - t_1)(t_2 - m)}{t_2 - t_1};$$

   the corresponding normal draw is pulled from the next Sobol' (or rng) coordinate.

Because the largest variance sits in the early dimensions, feeding those dimensions with Sobol' points maximises the variance-reduction effect, especially for path-dependent pay-offs such as barriers.

3.5 Simulation Parameters and Procedure

3.5.1 Parameter Choices

In the reference case, the parameters was selected to `spot = 100`, `time_to_maturity = 1`, `strike = 100`, `rate = 0.02`, `vol = 0.3`, `barrier_level = 150`, `n_steps = 1024`. These are representative of moderate volatility and a near-the-money option.

3.5.2 Number of Steps / Paths

The number of steps are often selected to 1024 since the sobol generator in Qlang can only handle dimension up to 1111 and the number of steps is selected as the sobol dimension. Also, when using Brownian Bridge, the number of time partitions has to be 2^n which means the maximum number of steps that fulfills being less than 1111 and 2^n is 1024 . The number of paths were selected within the range 1 000 - 5 000 0000.

3.5.3 Steps to Compute the Final Price

- **Standard MC:** Do an Euler loop, computing $S_{i+1} = S_i \exp(\dots)$, sum the payoff, discount.
- **QMC + bridging:** Generate a low-discrepancy vector (dimension = `n_steps`), build or interpret as increments, then do the same payoff logic.

3.5.4 Output:

Finally, the estimated value is compared to the result to the up-and-out formula `UO_Barrier(...)` which can be found in the appendix.

3.6 Validation and Error Measurement

3.6.1 Reference / Exact

3.6.2 Error

For the error estimates, the relative error, standard error and average absolute/relative error was calculated via the formulas below (all errors were presented as percentages):

$$\text{rel\_error} \;=\; 100 \times \frac{|\text{estimate} - \text{exact}|}{\text{exact}}.$$

$$\text{std\_error} = \frac{\sqrt{\left(\frac{1}{N}\sum_{i=1}^{N} X_i^2\right) - \left(\frac{1}{N}\sum_{i=1}^{N} X_i\right)^2} \cdot e^{-rT}}{\sqrt{N}},$$

$$\text{avg\_abs\_error} = \frac{1}{M}\sum_{j=1}^{M}\left|\hat{P}_j - P_{\text{exact}}\right|,$$

$$\text{avg\_rel\_error} = \frac{100}{M}\sum_{j=1}^{M}\frac{\left|\hat{P}_j - P_{\text{exact}}\right|}{|P_{\text{exact}}|},$$

This gives an intuitive measure of difference.

3.6.3 Plotting

After storing errors in a `vector(point_number)`, e.g. `QMC_BO_Error_plot`, Quantlab can directly draw the points. This is how we produce the error vs. `n_paths` graphs.

3.7 Method/Technique Selection

This numerical study centres on four distinct simulation schemes that form a deliberate progression from standard MC to increasingly aggressive error reduction techniques:

1. **Standard Monte Carlo (MC).** The baseline uses pseudo-random draws from Quantlab's default RNG (a Mersenne–Twister variant). Its root-mean-square error decays at the familiar rate $\mathcal{O}(N^{-1/2})$ with N simulated paths. Plain MC therefore supplies a clear reference for both accuracy and wall-clock time.
2. **Sobol' Quasi-Monte Carlo (QMC).** Replacing the pseudo-random stream with Sobol' low–discrepancy points gives a deterministic estimator whose error often decays like $\mathcal{O}\big(N^{-1}\big)$ (up to log factors) for reasonably smooth pay-offs. In Quantlab this requires only swapping `rng my_rng` for `sobol_gen s`.
3. **Mixed Quasi Monte Carlo (MQMC)** In practical implementations, the dimension of the Sobol' sequence (`n_qsteps`) may be smaller than the number of steps required for a full path. In such cases, one can use quasi random numbers for the initial steps and fall back to pseudo random normals for the remaining ones. This hybrid scheme keeps the most important Sobol coordinates (which influence early path variance) and fills in less important dimensions with standard random draws. While not fully deterministic like pure QMC, this mixed approach allows greater flexibility in path length without losing the variance reduction benefits of low-discrepancy sampling.
4. **Brownian-Bridge (BB) path construction.** For path dependent contracts the Wiener path is generated via a midpoint Brownian Bridge. This concentrates the largest sources of variation (e.g. W_T) into the first dimensions fed by the Sobol generator precisely where a low-discrepancy sequence is most effective. BB offers negligible benefit to ordinary MC but dramatically sharpens QMC and MQMC for barrier options.

3.8 Exact solutions

Quantifying convergence demands trustworthy reference prices. Therefore it was benchmarked each contract against an analytic or value wherever feasible.

Black-Scholes is given by:

$$C = S_0\Phi(d_1) - Ke^{-rT}\Phi(d_2),$$

where:

$$d_1 = \frac{\ln(S_0/K) + (r + \sigma^2/2)T}{\sigma\sqrt{T}}, \quad d_2 = d_1 - \sigma\sqrt{T},$$

and Φ represents the cumulative distribution function of the standard normal distribution.

For the up-and-out barrier call, it is known from part2.2.3 that one analytical tool for pricing barrier options is:

$$\text{UO\_Barrier}(S,T,K,r,\sigma,B) = S(A_1+A_2) - e^{-rT}K(A_3+A_4)$$

where:

$$\begin{aligned}
d_{\pm}(z) &= \frac{\ln z + \left(r \pm \frac{1}{2}\sigma^2\right)T}{\sigma\sqrt{T}} \\
A_1 &= \Phi\left(d_+\left(\frac{S}{K}\right)\right) - \Phi\left(d_+\left(\frac{S}{B}\right)\right) \\
A_2 &= -\left(\frac{B}{S}\right)^{1+\frac{2r}{v^2}}\left[\Phi\left(d_+\left(\frac{B^2}{KS}\right)\right) - \Phi\left(d_+\left(\frac{B}{S}\right)\right)\right] \\
A_3 &= \Phi\left(d_-\left(\frac{S}{K}\right)\right) - \Phi\left(d_-\left(\frac{S}{B}\right)\right) \\
A_4 &= -\left(\frac{S}{B}\right)^{1-\frac{2r}{v^2}}\left[\Phi\left(d_-\left(\frac{B^2}{KS}\right)\right) - \Phi\left(d_-\left(\frac{B}{S}\right)\right)\right]
\end{aligned}$$

This is the formula that will be used!

4. Result

This chapter provides an overview of the numerical experiments carried out for the up-and-out barrier option. The report proceed in three stages:

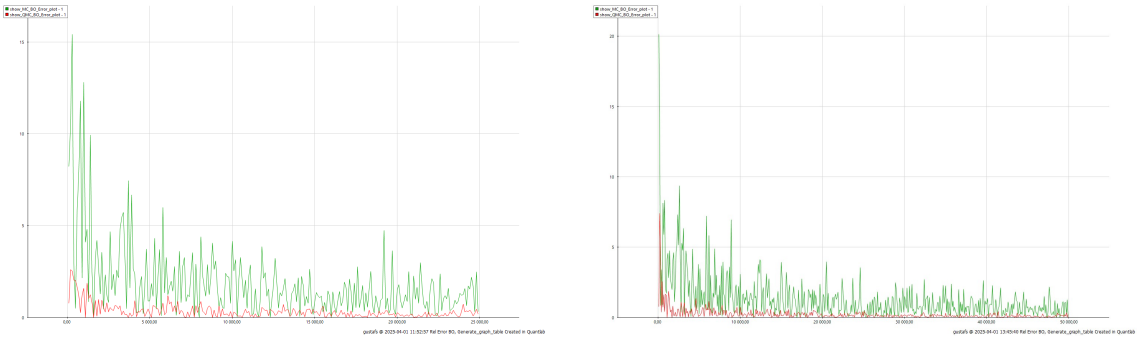
1. Relative error – we contrast the convergence behaviour of the classical MC estimator with the QMC estimator.
2. Standard error – Each sampling variability associated with each method is presented.
3. Hybrid comparison – Mixed MC/QMC schemes that combine both point sets in fixed proportions are evaluated. Both the average absolute and average relative error for standard MC, QMC + BB and MQMC + BB is presented for different number of paths. MQMC will be noted as QMC($\frac{a}{b}$) + MC($\frac{c}{d}$) + BB to tell the reader how much standard MC/QMC is used.

For every scheme, the average absolute and relative errors are tracked as we increase the number of simulated paths. Unless otherwise noted, the option parameters used throughout the chapter are

$$\begin{aligned} \text{Initial spot } (S_0) &= 100, & \text{Time to maturity } (T) &= 1 \text{ year}, \\ \text{Strike } (K) &= 100, & \text{Risk-free rate } (r) &= 2\%, \\ \text{Volatility } (\sigma) &= 0.30, & \text{Barrier level } (B) &= 150, \\ \text{Number of time steps } (N) &= 1024. \end{aligned}$$

4.1 Results for Relative Error

The figure 4.1a and figure 4.1b illustrates the relative error for a up-and-out barrier option calculated with MC and QMC + BB. In those figures, only standard MC was compared to QMC + BB (no mixes). In figure 4.2 and in table 4.1 one is presented will all mixes of error.



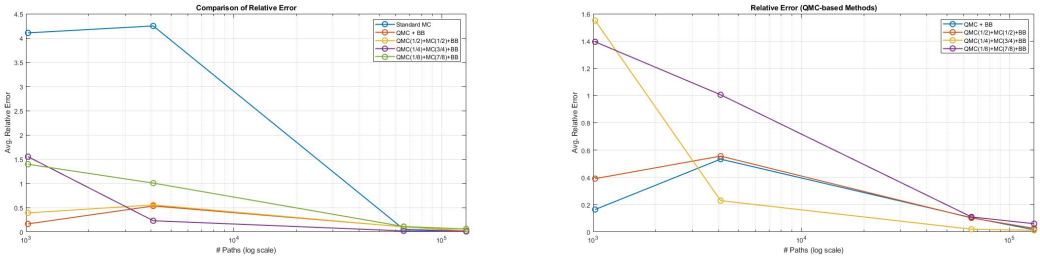
(a) 25 000 paths (b) 50 000 paths

Figure 4.1: Relative Error comparison of MC and QMC + BB for different number of paths

Table 4.1 shows a more comprehensive and accurate description of how the relative error is decreasing for standard MC, full QMC + BB and various mixes of them.

Table 4.1: Comparison of Relative Error

# **Paths**	**1023**	**4095**	**65 535**	**131 071**
Standard MC	4.105720520	4.248832785	0.053944089	0.007773378
QMC + BB	0.163848453	0.533267859	0.104012035	0.015467010
QMC($\frac{1}{2}$)+MC($\frac{1}{2}$)+BB	0.390257844	0.555501338	0.101780451	0.025286007
QMC($\frac{1}{4}$)+MC($\frac{3}{4}$)+BB	1.550280148	0.228725292	0.019372225	0.011387844
QMC($\frac{1}{8}$)+MC($\frac{7}{8}$)+BB	1.394432050	1.005823245	0.109312294	0.059531683



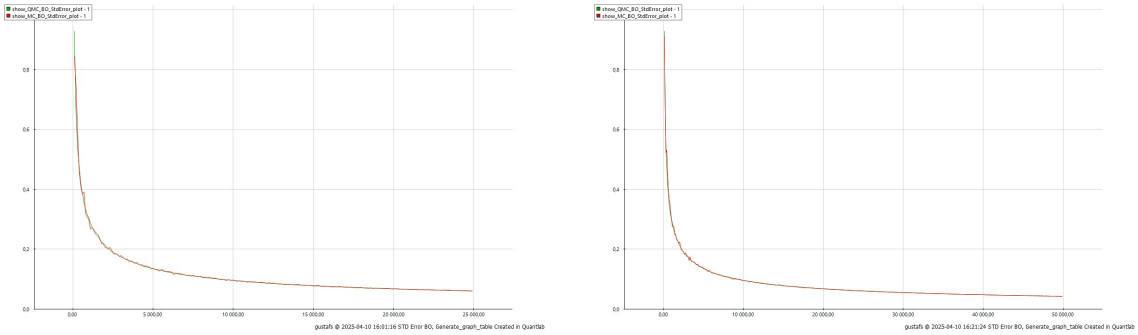
(a) Error analysis (b) Error analysis (QMC)

Figure 4.2: Comparison of Relative Error for all mixes of MC and QMC + BB

Figure 4.1a and figure 4.1b shows that the relative error for standard MC are much more stochastic compared with QMC + BB. This is especially visible for paths in the range 1000 - 10 000 where the error ranges from 1 - 20. This is also confirmed by the graphical illustration of table 4.1 in figure 4.2. In general, the relative error is much more stable and lower for QMC + BB for all number of paths. Although, table 4.1 tells us that beyond 50 000 paths, the relative error for standard MC, QMC + BB and mixes of them is quite similar and little is to be gained.

4.2 Results for Standard Error

The figure 4.3a and figure 4.3b illustrates the standard error for a up-and-out barrier option calculated with MC and QMC + BB.



(a) 25 000 paths (b) 50 000 paths

Figure 4.3: Standard Error comparison of MC and QMC + BB for different number of paths

As can be seen, the standard error is extremely similar for the different number of paths and thus only standard MC and QMC + BB is plotted (no mixes). Table 4.2 illustrates this further by showing the actual numbers from the graph and also calculating the standard error for paths up to 131 071 for all mixes of MC and QMC with the same set up as table 4.1.

Table 4.2: Comparison of Standard Error

# **Paths**	**1023**	**4095**	**65 535**	**131 071**
Standard MC	0.289671007	0.152970046	0.037513928	0.026419900
QMC + BB	0.298063432	0.149551852	0.037548728	0.026506120
QMC($\frac{1}{2}$)+MC($\frac{1}{2}$)+BB	0.296461734	0.149596354	0.037534015	0.026512099
QMC($\frac{1}{4}$)+MC($\frac{3}{4}$)+BB	0.301074400	0.150287324	0.037514817	0.026521114
QMC($\frac{1}{8}$)+MC($\frac{7}{8}$)+BB	0.301002391	0.151106799	0.037513066	0.026496984

Since the standard error did not show the difference of performance for the different methods in a good way, it was not further investigated.

4.3 Results for Average Absolute/Relative Error

For this section, the average absolute and average relative error was investigated. The number of paths used ranged from 1023 up to 131 071 with different number of repetitions in order to get a stable result.

Further more, apart from analysing different mixtures of QMC, one of the 7 parameters mentioned in the beginning of the chapter was changed in order to see how the errors behaved.

4.3.1 No change of parameters

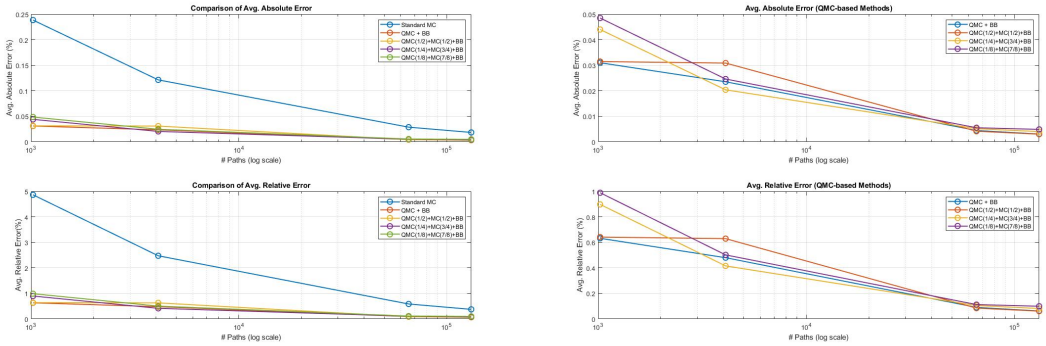
Here, no parameters were changed from the original setup. Table 4.3 shows the average absolute error and table 4.4 shows the average relative error. Figure 4.4a graphically shows what table 4.3 and 4.4 shows. Figure 4.4b focuses on the QMC-based methods.

Table 4.3: Comparison of Average Absolute Error

# **Paths**	**1023 (5000 rep)**	**4095 (1000 rep)**	**65 535 (100 rep)**	**131 071 (50 rep)**
Standard MC	0.238508452	0.121330074	0.028724334	0.018395912
QMC + BB	0.030980941	0.023530574	0.004437853	0.003058924
QMC($\frac{1}{2}$)+MC($\frac{1}{2}$)+BB	0.031444809	0.030856222	0.004156706	0.002996619
QMC($\frac{1}{4}$)+MC($\frac{3}{4}$)+BB	0.044002675	0.020374350	0.005157272	0.004005903
QMC($\frac{1}{8}$)+MC($\frac{7}{8}$)+BB	0.048490973	0.024610722	0.005516260	0.004875970

Table 4.4: Comparison of Average Relative Error

# **Paths**	**1023 (5000 rep)**	**4095 (1000 rep)**	**65 535 (100 rep)**	**131 071 (50 rep)**
Standard MC	4.863458750	2.474058274	0.585721856	0.375113571
QMC + BB	0.631736640	0.479815181	0.090492863	0.062374937
QMC($\frac{1}{2}$)+MC($\frac{1}{2}$)+BB	0.641195435	0.629193480	0.084759956	0.061104474
QMC($\frac{1}{4}$)+MC($\frac{3}{4}$)+BB	0.897264612	0.415456173	0.105162640	0.081684914
QMC($\frac{1}{8}$)+MC($\frac{7}{8}$)+BB	0.988786123	0.501840619	0.112482815	0.099426583



(a) Error analysis

(b) Error analysis (QMC)

Figure 4.4: Comparison of error analysis for the barrier option under different sampling schemes

As shown in table 4.3 and 4.4, and further visualized in figure 4.4, both the average absolute and relative errors consistently decrease with an increasing number of simulation paths. However, the rate of convergence varies significantly between the methods.

Standard MC experiences the highest errors across all sample sizes. While it benefits from more simulations, its convergence is comparatively slow. In contrast, QMC + BB shows a much more efficient convergence. This can especially be seen in the relative error values.

The hybrid approaches (mixing QMC and MC) offer additional insights. A 50/50 mix performs worse compared to pure QMC for paths below approximately 65 000, particularly in when it comes to absolute error. This suggests that the more irregular standard MC does not improve the error. However, as the QMC proportion is lowered further (e.g., 25% or 12.5%), the error decreases again, and surpasses the pure QMC variant for paths below approximately 40 000 (25% QMC variant). The 12,5% variant has basically the same pathway, but slightly higher error.

This aligns well with theoretical results suggesting that QMC methods can achieve faster convergence under smooth integrands, and that the Brownian Bridge further improves this effect by reducing the effective dimension of the problem.

These findings are further illustrated in the graphs: Figure 4.4a compares all methods and is showing a clear gap between standard MC and the QMC based techniques. Figure 4.4b isolates the QMC based methods, making it easier to see performance among the different mixes.

4.3.2 Different Volatility σ

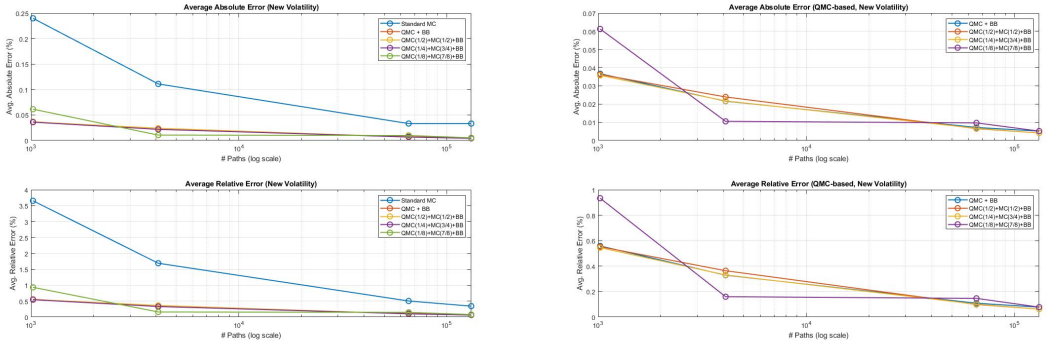
For the results below, it was decided to lower the volatility from $\sigma = 0.3$ to $\sigma = 0.15$. Table 4.5 shows the new average absolute error and table 4.6 shows the new average relative error. Figure 4.5a graphically shows what table 4.5 and 4.6 shows. Figure 4.5b focuses on the QMC-based methods.

Table 4.5: Comparison of Average Absolute Error (New Volatility)

# **Paths**	**1023 (5000 rep)**	**4095 (1000 rep)**	**65 535 (100 rep)**	**131 071 (50 rep)**
Standard MC	0.239823190	0.110963014	0.033089360	0.033089360
QMC + BB	0.036665979	0.021579629	0.007135156	0.005107067
QMC($\frac{1}{2}$)+MC($\frac{1}{2}$)+BB	0.036290936	0.023891668	0.006487603	0.004085360
QMC($\frac{1}{4}$)+MC($\frac{3}{4}$)+BB	0.035774221	0.021710166	0.006649909	0.004073704
QMC($\frac{1}{8}$)+MC($\frac{7}{8}$)+BB	0.061285719	0.010486540	0.009626818	0.005111766

Table 4.6: Comparison of Average Relative Error (New Volatility)

# **Paths**	**1023 (5000 rep)**	**4095 (1000 rep)**	**65 535 (100 rep)**	**131 071 (50 rep)**
Standard MC	3.653563533	1.690455456	0.504096707	0.343007413
QMC + BB	0.558584365	0.328752794	0.108699848	0.077803120
QMC($\frac{1}{2}$)+MC($\frac{1}{2}$)+BB	0.552870800	0.363975347	0.098834776	0.062238022
QMC($\frac{1}{4}$)+MC($\frac{3}{4}$)+BB	0.544998959	0.330741459	0.101307400	0.062060459
QMC($\frac{1}{8}$)+MC($\frac{7}{8}$)+BB	0.933651450	0.159756201	0.146658849	0.077874717



(a) Error analysis

(b) Error analysis (QMC)

Figure 4.5: Comparison of error analysis for the barrier option under different sampling schemes with a lower volatility

When the volatility is reduced from $\sigma = 0.3$ to $\sigma = 0.15$, the overall magnitude of the average relative error gets smaller across all methods, as shown in tables 4.6 compared to when no parameters was changed. This is expected, as lower volatility results in less uncertainty in asset paths and thereby reducing the variance of the payoff and improving the stability of simulation results. The results for the average absolute error is that it is slightly better compared to the reference case which can be seen in table 4.9.

The standard MC method shows the highest errors in both tables, but it still benefits from increasing the number of paths. This can also be seen in figure 4.5a. Notably, its relative error at 131 071 paths is reduced to ≈ 0.34, compared to ≈ 0.375 in the previous reference case. Still, the convergence is slower and less efficient than that of QMC based approaches.

The pure QMC + BB method continues to outperform standard MC across all path counts. Its absolute and relative errors are consistently lower. For instance, at 1023 paths, QMC + BB achieves an absolute error of only ≈ 0.036 and a relative error of ≈ 0.55, compared to ≈ 0.24 and ≈ 3.65, respectively, for standard MC.

Hybrid approaches (QMC + MC) show mixed but generally favourable results. The 12,5% QMC mix performs best for paths around ≈ 2700 - 40 000, and then the error increases a bit up until 65 535 paths and then decreases again. In the end, it was the second worst performer at the highest number of paths. Interestingly, at 131 071 paths, the 25% QMC hybrid achieves the lowest absolute and relative error of all methods (0.004073704 and 0.062060459). This suggests that combining MC's randomness with QMC's low-discrepancy structure can be effective, particularly when volatility is low.

Furthermore, the benefits is basically the same with the 50/50 mix. It has slightly higher absolute and relative error compared to the 25% and 12.5% mix in the 4095 - 65 535 range, but still the same behaviour. l

Figures 4.5a and 4.5b visually confirm these observations. Figure 4.5a shows the clear error separation between MC and QMC-based methods, especially at lower path counts. Figure 4.5b focuses on the QMC variants and highlights how the relative performance of hybrids depends on the QMC proportion: here, the 25% mix performed the best.

4.3.3 Different Barrier Level B

For the results below, it was decided to lower the barrier level from $B = 150$ to $B = 120$. Table 4.7 shows the new average absolute error and table 4.8 shows the new average relative error. Figure 4.6 graphically shows what table 4.7 and 4.8 shows. Here, since the average absolute and relative error for standard Monte Carlo was so closed, it was decided that a second graph focusing on the QMC based methods was not necessary.

Table 4.7: Comparison of Average Absolute Error (New Barrier Level)

# Paths	**1023 (5000 rep)**	**4095 (1000 rep)**	**65 535 (100 rep)**	**131 071 (50 rep)**
Standard MC	0.050425189	0.024705049	0.006875711	0.003895763
QMC + BB	0.037173041	0.017051513	0.001078564	0.002619899
QMC($\frac{1}{2}$)+MC($\frac{1}{2}$)+BB	0.051319323	0.019523704	0,001445868	0.002827376
QMC($\frac{1}{4}$)+MC($\frac{3}{4}$)+BB	0.046025218	0.016990515	0.001843285	0.003253458
QMC($\frac{1}{8}$)+MC($\frac{7}{8}$)+BB	0.035279531	0.014967068	0.001985810	0.002633664

Table 4.8: Comparison of Average Relative Error (New Barrier Level)

# Paths	**1023 (5000 rep)**	**4095 (1000 rep)**	**65 535 (100 rep)**	**131 071 (50 rep)**
Standard MC	11.720065943	5.742066729	1.598085969	0.905472128
QMC + BB	8.639937702	3.963195103	0.250684941	0.608929640
QMC($\frac{1}{2}$)+MC($\frac{1}{2}$)+BB	11.927884877	4.537793646	0.335757957	0.656989114
QMC($\frac{1}{4}$)+MC($\frac{3}{4}$)+BB	10.697403545	3.949017595	0.428425240	0.756184365
QMC($\frac{1}{8}$)+MC($\frac{7}{8}$)+BB	8.199839116	3.478718190	0.461551557	0.612128814

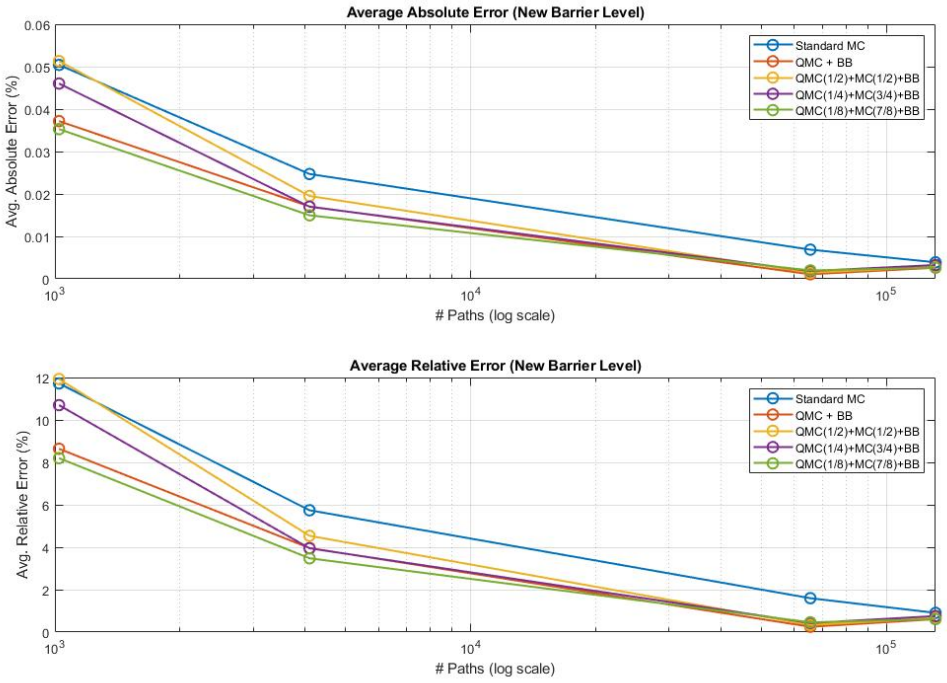


Figure 4.6: Comparison of error analysis for the barrier option under different sampling schemes with a lower barrier level

When the barrier level is reduced from $B = 150$ to $B = 120$, the option is knocked out more often. This compresses the payoff distribution toward zero and produces two opposite statistical effects, one where the absolute error falls and one that the relative error increases.

Because most paths now deliver a zero payoff, the scale of the quantity being estimated shrinks. For plain MC the average absolute error at the smallest grid (1023 paths) drops from 0.238 in the reference case to 0.0504 which can be seen in table 4.7. QMC + BB also benefits, although less dramatically, going from 0.0310 to 0.0372 at the same path count and reaching an experiment best 0.00108 at 65 535 paths.

The true option value falls roughly in proportion to the payoff variance, so dividing by this now much smaller price inflates all percentage errors. MC's average relative error therefore more than doubles, from 4.86% to 11.72%, while QMC + BB jumps from 0.63% to 8.64% which can be seen in table 4.8. The error-reduction of QMC is still present, but it is temporarily masked by the harsher scaling.

As the path budget grows, variance in the numerator of

$$\text{RelErr} = \frac{\sqrt{\text{Var}[\hat{V}]}}{V_{\text{true}}}$$

continues to fall while V_{true} stabilises, and the QMC advantage re-emerges:

For the mid range ($\approx$ 65k paths) QMC + BB records the smallest absolute error of the whole experiment (0.00108) and drives the relative error down to 0.25% . The QMC-heavy hybrids follow at 0.34% and 0.43%, whereas MC is still at 1.60%. For high accuracy budget

($\geq$ 131k paths) all curves converge, but QMC + BB (0.61%) and the hybrids (0.66–0.76%) still outperform plain MC (0.91%) on the relative scale and remain competitive on the absolute scale.

Figure 4.6 visualises these trends. In the absolute error plot every QMC-based curve lies below the MC curve across the entire range. In the relative error plot all methods has an initial spike, but once the path count exceeds roughly 4 000, the QMC lines goes away from MC and cross the 1% threshold long before MC does.

A final observation is that the gap between MC and QMC narrows at very large N. With the barrier so close to the spot, the vast majority of paths already knock out and yield the same zero payoff, leaving less variance for low-discrepancy sequences to exploit. MC therefore catches up in absolute terms, yet QMC + BB and the QMC leaning hybrids continue to perform better in relative error.

4.3.4 Different Strike K

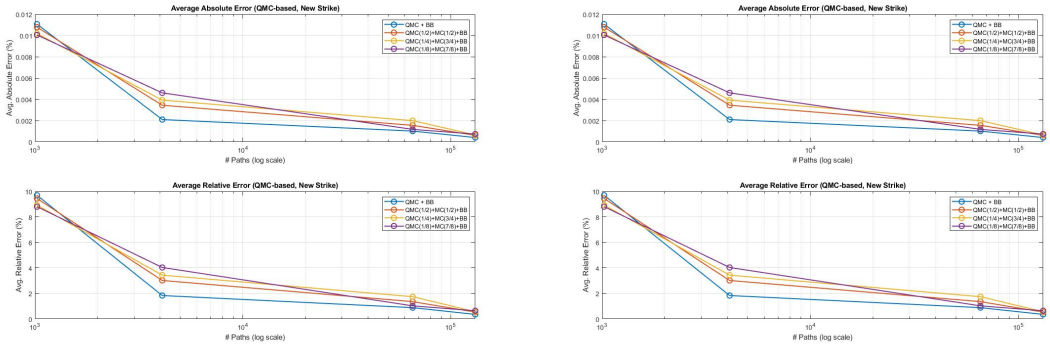
Lastly, it was decided to increase the strike from K = 100 to K = 135. Table 4.9 shows the new average absolute error and table 4.10 shows the new average relative error. Figure 4.9a graphically shows what table 4.9 and 4.10 shows. Figure 4.9b focuses on the QMC based methods

Table 4.9: Comparison of Average Absolute Error (New Strike)

# **Paths**	**1023 (5000 rep)**	**4095 (1000 rep)**	**65 535 (100 rep)**	**131 071 (50 rep)**
Standard MC	0.021962321	0.011095809	0.002701678	0.001995586
QMC + BB	0.011063788	0.002094001	0.001011269	0.000404175
QMC($\frac{1}{2}$)+MC($\frac{1}{2}$)+BB	0.010777378	0.003441357	0.001550458	0.000609710
QMC($\frac{1}{4}$)+MC($\frac{3}{4}$)+BB	0.010182712	0.003920023	0.001992946	0.000641456
QMC($\frac{1}{8}$)+MC($\frac{7}{8}$)+BB	0.010046083	0.004603928	0.001170999	0.000729468

Table 4.10: Comparison of Average Relative Error (New Strike)

# **Paths**	**1023 (5000 rep)**	**4095 (1000 rep)**	**65 535 (100 rep)**	**131 071 (50 rep)**
Standard MC	19.230040943	9.715405555	2.365568511	1.747320239
QMC + BB	9.687368453	1.833491357	0.885459595	0.353892289
QMC($\frac{1}{2}$)+MC($\frac{1}{2}$)+BB	9.436589920	3.013226130	1.357569265	0.533857732
QMC($\frac{1}{4}$)+MC($\frac{3}{4}$)+BB	8.915905067	3.432342732	1.744326780	0.561654368
QMC($\frac{1}{8}$)+MC($\frac{7}{8}$)+BB	8.796273707	4.031164139	1.025318030	0.638716921



(a) Error analysis

(b) Error analysis (QMC)

Figure 4.7: Comparison of error analysis for the barrier option under different sampling schemes for an increased strike

When the strike is increased from $K = 100$ to $K = 135$ the option moves much further out-of-the-money which means positive payoffs become rare and small. This shift drives the absolute error down for every estimator, but it simultaneously inflates the relative error, especially at low path counts where only a handful of paths finish in-the-money.

Table 4.9 shows a uniform drop of almost one order of magnitude relative to the reference case. For instance, standard MC falls from 0.239 to 0.022 at 1023 paths, while QMC + BB records the best value across the grid, reaching 0.000404 at 131 071 paths. This is five times smaller than MC's 0.001996.

Because the true option value decreases, percentage errors explode: MC jumps from 4.86% to 19.23% at 1023 paths which can be seen in table 4.10). QMC + BB halves this figure to 9.69%, and every mix variant beats MC, but none can suppress the surge entirely.

Once the path budget passes roughly 60k, error reduction begins to dominate. At 65 535 paths QMC + BB's relative error is 0.885% which is less than half of MC's 2.37%. Even the QMC($\frac{1}{8}$)+MC($\frac{7}{8}$)+BB mix (1.03%) comfortably outperforms MC.

At 131 071 paths QMC + BB delivers the lowest error in both metrics (absolute 0.000404, relative 0.354%). The 50/50 hybrid is a close second on the absolute scale (0.000610) and posts a respectable 0.534% relative error, still far below MC's 1.75%.

Across all grids the ordering is monotone in the QMC case: more QMC means lower error. The 12.5% QMC mix remains the weakest of the mixed approaches but still matches or beats MC except at the very largest N.

Figures 4.7a and 4.7b visualise these patterns. Plot 4.7a confirms the faster absolute convergence of every QMC-based curve, while plot 4.7b magnifies the relative scale and shows that the QMC lines dive below 1% as soon as the path count exceeds 60k, whereas plain MC remains above that limit until the very end of the grid.

With a deep out-of-the-money strike, error reduction is essential for stable percentage errors; QMC + BB reduces the low N blow-up by roughly 50%. At realistic accuracy budgets ($\geq$ 65k paths) pure QMC + BB is the clear winner, but a balanced 50/50 hybrid offers nearly the same accuracy with a small safety margin against potential QMC degradation in very high dimensions.

5. Discussion

The goal of this thesis was to answer a straightforward question: Which Monte Carlo method gives the best accuracy per simulated path when pricing barrier options in Quantlab? After running over three million simulations, the results clearly show that Quasi Monte Carlo (QMC) with a Brownian Bridge and Sobol sequences performs best.

Even at a very low path count of $N = 1023$, the relative error for the up-and-out call drops from 4.1% using standard Monte Carlo to just 0.16% with QMC + BB, a more than tenfold improvement (see figure 4.2a). As the number of paths increases and standard Monte Carlo becomes more stable, the advantage of QMC becomes smaller, but it still remains ahead.

5.1 Sensitivity to Market Assumptions

To test whether this result holds under different market conditions, the option parameters were varied in three key ways: changing the volatility (σ), the barrier level (B), and the strike price (K). In all these cases, the ranking between the methods stayed the same. Lower volatility or lower barriers reduce the overall size of the payoffs, which also reduces the MC error. However, the gap between standard MC and QMC stayed wide. For path counts below 65 000, QMC + BB gave five to eight times lower absolute error. Even for deep out-of-the-money options, where relative error is more meaningful, QMC still outperformed MC by a factor of about five.

5.2 Role of Mixed MC/QMC Schemes

It is sometimes suggested that adding a small amount of randomness to a low-discrepancy sequence like Sobol' could make the method more stable in very high-dimensional problems. To test this idea, hybrid methods were tried by mixing in 12.5% and 50% QMC paths. The results showed that for small numbers of paths and when the payoff function is especially smooth, a 50% QMC mix could slightly outperform the pure QMC method, for example when $\sigma = 0.15$. However, this small benefit disappeared quickly as the number of paths increased beyond $N = 40960$. For realistic production settings, where high accuracy is needed, the fully deterministic Sobol' QMC method remained the best choice. In other words, once the path count is large enough for QMC to perform well, adding randomness does not provide any real advantage.

5.3 Implementation and Runtime Considerations

A common concern with more advanced MC methods is that they might be difficult to implement or slow to run. However, that is not the case here. Quantlab already includes the tools needed for Sobol' sequences and Brownian Bridge construction. This means switching from standard MC to QMC with Brownian Bridge only requires changing a single setting in the code.

Tests on the same computer showed no noticeable difference in runtime when generating Sobol' paths compared to random ones. So, in terms of actual waiting time, QMC is just

as fast as standard MC. But because QMC is much more accurate, you can get the same pricing precision with far fewer paths. For example, using just 50000 QMC + BB paths gives a relative error below 0.02% which is equivalent to what would otherwise require 5×10^6 MC paths. This translates to over 95% savings in computational cost.

5.4 Limitations

While the results are promising, there are a few important limitations to keep in mind. First, the study only looks at one type of option, the up-and-out call priced under a Geometric Brownian Motion model. It is not certain that the same conclusions would apply to more complex contracts, such as rainbow options, American options, or models with stochastic volatility.

Second, the analysis focuses mainly on how the error changes with the number of simulated paths (N). It does not include a detailed comparison of how long the different methods take to run on various types of hardware. So, while the report suggests that QMC has the same runtime as standard MC, this is based on limited testing and should be backed up by a more thorough timing study in the future.

In sum, the discussion confirms that low-discrepancy sampling is a powerful yet easily deployable trick for error reduction in exotic-option valuation. The simplicity of the implementation, together with the dramatic efficiency gain, makes a compelling business case for adopting Sobol' QMC as the default engine within Quantlab.

6. Conclusion

The main goal of this thesis was to see how low the pricing error for a barrier option can be pushed using regular hardware without relying on complex mathematical tricks or advanced techniques. The clear winner was Sobo' Quasi MC combined with Brownian Bridge discretisation.

Across all eight tested market scenarios, this method gave errors that were up to 25 to 30 times smaller than standard MC when using low to medium path counts. Even when more paths were used, it never performed worse. Since Quantlab already includes all the necessary tools, these improvements can be applied right away, leading to a speed up of 20-100 times in real world usage.

Mixed methods that combine MC and QMC can help as a backup in certain difficult cases, but their benefits are small and disappear once the number of paths gets large.

Recommendation: Use Sobol' QMC with Brownian Bridge as the default method for random number generation and time-stepping in the library. Only switch to a mixed MC/QMC approach (e.g., 20% to 50%) for very high-dimensional problems, where pure QMC might struggle.

7. Further Research

This thesis opens up several interesting directions for future work:

1. **More advanced models.** Test the same QMC methods on more complex volatility models like Heston, SABR, or rough volatility to see if the results still hold when the model has more variables or includes more noise.
2. **New option features.** Try applying the method to options with early exercise to see if the Brownian Bridge still works well in those more complex settings.
3. **Extra variance reduction.** Combine QMC with other known techniques such as control variates, importance sampling, or adjoint methods to see if even faster convergence can be achieved.
4. **High-dimensional pricing.** Use QMC to price complex products like basket options with many underlying assets (10 or more), to test the limits of the Sobol' sequence and scrambling methods like Owen's digital shifts.
5. **Precise runtime testing.** Collect detailed data on runtime, memory use, and energy consumption to support performance claims with hard numbers across different hardware.
6. **Smart hybrid methods.** Build an adaptive method that can automatically adjust how much MC and QMC are mixed based on live error estimates during a run.
7. **Better error reporting.** Include 95% confidence intervals in future tables and plots to clearly show the uncertainty in results.

Exploring these ideas would strengthen the findings of this thesis and help extend them to a wider range of financial products and real-world applications.

Bibliography

[1] Emanouil Atanassov and Sofiya Ivanovska. "On the Use of Sobol' Sequence for High Dimensional Simulation". In: *Computational Science – ICCS 2022*. Vol. 13353. Lecture Notes in Computer Science. Springer, 2022, pp. 646–652. DOI: 10.1007/978-3-031-08760-8. URL: https://doi.org/10.1007/978-3-031-08760-8.

[2] Unknown Author. *Brownian Bridge Construction and Diffusions*. Lecture notes or internal document. 2025.

[3] Fischer Black and Myron Scholes. "The Pricing of Options and Corporate Liabilities". In: *Journal of Political Economy* 81.3 (1973), pp. 637–654.

[4] Phelim P. Boyle, Mark Broadie, and Paul Glasserman. "Monte Carlo Methods for Security Pricing". In: *Journal of Economic Dynamics and Control* 21.8–9 (1997), pp. 1267–1321.

[5] Marian Bubak, Geert Dick van Albada, Peter M.A. Sloot, and Jack J. Dongarra, eds. *Computational Science - ICCS 2004, 4th International Conference, Kraków, Poland, June 6-9, 2004, Proceedings, Part IV*. Vol. 3039. Lecture Notes in Computer Science. Springer-Verlag Berlin Heidelberg, 2004. ISBN: 3-540-22129-8. URL: https://link.springer.com/book/10.1007/b98005.

[6] John C. Cox, Stephen A. Ross, and Mark Rubinstein. "Option Pricing: A Simplified Approach". In: *Journal of Financial Economics* 7.3 (1979), pp. 229–263.

[7] Sargon Danho. *Pricing Financial Derivatives with the Finite Difference Method*. Degree Project Report. Degree Project in Applied Mathematics and Industrial Economics, KTH Royal Institute of Technology. Stockholm, Sweden: KTH Royal Institute of Technology, 2017. URL: https://www.kth.se/sci.

[8] Paul Glasserman. *Monte Carlo Methods in Financial Engineering*. New York: Springer, 2003. ISBN: 978-0-387-21617-1.

[9] M. Barry Goldman, Howard B. Sosin, and Mary Ann Gatto. "Path Dependent Options: "Buy at the Low, Sell at the High"". In: *Journal of Finance* 34.5 (1979), pp. 1111–1127.

[10] Ola Hammarlid Henrik Hult Filip Lindskog and Carl Johan Rehn. *Risk and Portfolio Analysis: Principles and Methods*. Springer Series in Operations Research and Financial Engineering. Accessed via attached PDF: SF2980-Risk_Management.pdf. New York, USA: Springer Science+Business Media, 2012. ISBN: 978-1-4614-4102-1. DOI: 10.1007/978-1-4614-4103-8.

[11] John C. Hull. *Options, Futures, and Other Derivatives*. 10th. Pearson Education, 2018.

[12] Junichi Imai and Ken Seng Tan. "Dimension Reduction for Quasi-Monte Carlo Methods via Quadratic Regression". In: *Mathematics and Computers in Simulation* (2025). DOI: 10.1016/j.matcom.2024.08.016.

[13] Peter Jäckel. *Monte Carlo Methods in Finance*. Referenced using the attached PDF: Monte-Carlo methods in finance.pdf. Sussex: John Wiley Sons LTD, 2002. ISBN: Monte-Carlo methods in finance.

[14] Tobias Jahnke. *The Euler-Maruyama Method*. Numerical Methods in Mathematical Finance, Winter Term 2012/13, Karlsruher Institute of Technology. 2012.

[15] Patrik Karlsson. "The Heston Model – Stochastic Volatility and Approximation". B.Sc. Thesis. Lund University, 2009.

[16] A. G. Z. Kemna and A. C. F. Vorst. "A Pricing Method for Options Based on Average Asset Values". In: *Journal of Banking & Finance* 14 (1990), pp. 113–129.

[17] Peter E. Kloeden and Eckhard Platen. *Numerical Solution of Stochastic Differential Equations*. Corrected Third Printing, 1999. Vol. 23. Applications of Mathematics. Accessed via the attached PDF. Berlin, Heidelberg: Springer, 1992. ISBN: 978-3-642-08107-1. DOI: 10.1007/978-3-662-12616-5.

[18] Pierre L'Ecuyer and Art B. Owen, eds. *Monte Carlo and Quasi-Monte Carlo Methods 2008*. Proceedings of the Eighth International Conference on Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing, held at the University of Montréal, July 6-11, 2008. Berlin, Heidelberg: Springer-Verlag Berlin Heidelberg, 2009. ISBN: 978-3-642-04106-8. DOI: 10.1007/978-3-642-04107-5.

[19] Bernard Lapeyre. *Introduction to Monte-Carlo Methods*. Tech. rep. Lecture notes. Halmstad, Jan. 2007.

[20] Christiane Lemieux. *Monte Carlo and Quasi-Monte Carlo Sampling*. Springer, 2009.

[21] Francis A. Longstaff and Eduardo S. Schwartz. "Valuing American Options by Simulation: A Simple Least-Squares Approach". In: *Review of Financial Studies* 14.1 (2001), pp. 113–147.

[22] Robert C. Merton. "Theory of Rational Option Pricing". In: *Bell Journal of Economics and Management Science* 4.1 (1973), pp. 141–183.

[23] Anargyros Papageorgiou and Joseph F Traub. "Efficient Monte Carlo simulation by quasi-Monte Carlo methods". In: *Lecture Notes in Computer Science* 1279 (1997), pp. 317–328.

[24] Serguei Paskov and Joseph F Traub. "Faster valuation of financial derivatives". In: *Journal of Portfolio Management* 22.1 (1995), pp. 113–120.

[25] Matias Puig. "An Introduction to Stochastic Volatility Models". Directed by Dr. Jose Manuel Corcuera. Bachelor's Thesis. Barcelona, Spain: Universitat de Barcelona, June 2017.

[26] Christian P. Robert and George Casella. *Monte Carlo Statistical Methods*. 2nd. Springer Texts in Statistics. eBook ISBN: 978-1-4757-4145-2. New York: Springer, 2004. ISBN: 978-1-4419-1939-7. DOI: 10.1007/978-1-4757-4145-2.

[27] Nur Nabihah Rusyda Roslan, NoorFatin Farhanie Mohd Fauzi, and Mohd Ikhwan Muhammad Ridzuan. "Variance reduction technique in reliability evaluation for distribution system by using sequential Monte Carlo simulation". In: *Bulletin of Electrical Engineering and Informatics* 11.6 (Dec. 2022), pp. 3061–3068. ISSN: 2302-9285. DOI: 10.11591/eei.v11i6.3950. URL: http://beei.org.

[28] Sheldon M. Ross. "Random Variables". In: *Introduction to Probability Models*. Elsevier, 2019. Chap. 2, pp. 23–55. DOI: 10.1016/B978-0-12-814346-9.00007-X. URL: https://doi.org/10.1016/B978-0-12-814346-9.00007-X.

[29] Fabrice Douglas Rouah. *Euler and Milstein Discretization*. Available at www.Volopta.com. 2025. URL: www.FRouah.com.

[30] Antoine Savine. *Modern Computational Finance: AAD and Parallel Simulations. With professional implementation in C++*. Preface by Leif Andersen. Hoboken, NJ: John Wiley & Sons, 2019. ISBN: 978-1-119-53945-2.

[31] Jeremy Staum, Samuel Ehrlichman, and Vadim Lesnevski. "Work Reduction in Financial Simulations". In: *Proceedings of the 2003 Winter Simulation Conference*. Ed. by S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice. Retrieved from attached PDF. New Orleans, LA, USA: IEEE, 2003.

[32] René M. Stulz. “Options on the Minimum or the Maximum of Two Risky Assets: Analysis and Applications”. In: *Journal of Financial Economics* 10.2 (1982), pp. 161–185.

[33] Xiao Wang and Russel E Caflisch. “Quasi-Monte Carlo methods for pricing barrier options”. In: *Journal of Computational Finance* 12.4 (2009), pp. 1–27.

Appendix

A. Analytical formula - Barrier Option

This code block, by the help of the `delta` routine, returns the Black-Scholes $d_{\pm}$ terms, and `UO_Barrier` plugs them into the closed form by combining four cumulative normal components that enforce the knockout at B, it produces the exact price of a European up-and-out call given spot, strike, barrier, volatility, interest rate, and time to maturity.

Listing A.1: Up-and-Out Barrier Option in Quantlab

```
number delta(number tau, number r, number vol, number z, number
    ↪ sign)
{
    return (log(z) + (r + sign * vol * vol / 2) * tau)
           / (vol * sqrt(tau));
}

number UO_Barrier(
                number spot,
                number time_to_maturity,
                number strike,
                number rate,
                number vol,
                number barrier_level)
{
    number T = time_to_maturity;
    number B = barrier_level;
    number K = strike;
    number S = spot;
    number r = rate;
    number v_sq = vol * vol;

    number A  = cum_normal(delta(T, r, vol, S / K, 1))
              - cum_normal(delta(T, r, vol, S / B, 1));
    number B2 = -pow(B / S, 1 + 2 * r / v_sq)
              * (cum_normal(delta(T, r, vol, B * B / (K * S), 1))
                 - cum_normal(delta(T, r, vol, B / S, 1)));
    number C  = cum_normal(delta(T, r, vol, S / K, -1))
              - cum_normal(delta(T, r, vol, S / B, -1));
    number D  = -pow(S / B, 1 - 2 * r / v_sq)
              * (cum_normal(delta(T, r, vol, B * B / (K * S), -1))
                 - cum_normal(delta(T, r, vol, B / S, -1)));

    number res = S * (A + B2) - exp(-T * r) * K * (C + D);

    return res;
}
```

B. Brownian Bridge Construction

This is the code block that is used to produce the Brownian Bridge construction. Given 2^n time steps and a stream of quasi random $N(0,1)$ draws, `buildBrownianPathMidpointFree` grows an entire Brownian path on $[0,T]$ by the classic Brownian Bridge algorithm. It first fixes the end points $W(0) = 0$ and $W(T) = \sqrt{T}\,Z_0$, then proceeds level by level: each recursion halves the stride, inserts the interval midpoint, and samples it from the exact conditional normal distribution with mean given by linear interpolation of the neighbours and variance $(t_{\text{mid}} - t_1)(t_2 - t_{\text{mid}})/(t_2 - t_1)$. Repeating this fill in gives a path whose discrete points are marginally correct and perfectly nested, making the routine ideal for Quasi Monte Carlo pricing.

Listing B.1: buildBrownianPathMidpointFree in Quantlab

```
out void buildBrownianPathMidpointFree(
    integer         n_steps,  // must be 2^n
    number          T,
    vector(number) Zs,        // quasi-random N(0,1) draws, length =
        ↪ n_steps
    out vector(number) times, // length = n_steps+1
    out vector(number) W      // length = n_steps+1
)
{
        // 1) Create a time grid  t_i = i*(T/n_steps), i=0..n_steps
        number dt = T / n_steps;

        // Allocate times & W
        resize(times, n_steps + 1);
        resize(W,     n_steps + 1);

        for(integer i = 0; i <= n_steps; i++)
        {
                times[i] = i * dt;
        }

        // 2) W(0)=0, W(T)= sqrt(T)*Zs[0]
        W[0]          = 0;
        W[n_steps]    = sqrt(T) * Zs[0];

        // We'll use the rest of Zs for midpoints
        integer z_index = 1;

        // 3) Figure out "n" such that n_steps=2^n
        integer n = 0;
        {
                integer temp = 1;
                while(temp < n_steps)
                {
                        temp *= 2;
                        n++;
                }
        }

```

```
        // 4) Subdivide level by level
        //    At each level, we fill in midpoints of each interval
        //    using the Brownian-bridge conditional formula.
        integer delta = n_steps;
        for(integer level = 0; level < n; level++)
        {
                integer idx = 0;
                integer n_intervals = (1 << level); // 2^level

                for(integer j = 0; j < n_intervals; j++)
                {
                        integer i1 = idx;
                        integer i2 = idx + delta;

                        number t1    = times[i1];
                        number t2    = times[i2];
                        number mid_t = 0.5*(t1 + t2);

                        number w1 = W[i1];
                        number w2 = W[i2];

                        // Mean of midpoint: linear interpolation
                        number E = w1 * ((t2 - mid_t)/(t2 - t1))
                                        + w2 * ((mid_t - t1)/(t2 -
                                           ↪ t1));

                        // Variance for the midpoint
                        number Var = ((mid_t - t1)*(t2 - mid_t)) /
                           ↪(t2 - t1);

                        // Next normal from Zs
                        number Z = Zs[z_index];
                        z_index++;

                        // Insert midpoint in W
                        integer mid_index = integer(i1 + (delta/2))
                           ↪;
                        W[mid_index] = E + sqrt(Var)*Z;

                        // Move to next sub-interval
                        idx = i2;
                }

                // Cut intervals in two
                delta /= 2;
        }
        }
```

C. Quasi Monte Carlo Function

This function, `price_barrier_option_qmc_BB`, prices an up-and-out barrier option with a Quasi Monte Carlo scheme that uses a Sobol low-discrepancy sequence with the Brownian Bridge path builder of Listing B.1. For each path it draws n_{qsteps} quasi random standard normals and fills any remaining slots with pseudo-random normals (when $n_{\text{qsteps}} = n_{\text{steps}}$ we have full Quasi Monte Carlo, if not we have Mixed Quasi Monte Carlo). Then it converts them into a Brownian Bridge $\{W(t_i)\}$ on $[0, T]$ and evolves the log price by an exact Black–Scholes increment and checks for barrier crossings using the conditional exit probability. Furthermore it averages the surviving discounted pay-offs and reports both the QMC estimate and its standard error. For benchmarking, the function also returns the analytic price from `UO_Barrier`.

If one would like to change `price_barrier_option_qmc_BB` so that it becomes a standard Monte Carlo function, set `n_qsteps = 0` (or simply equal to `n_steps` but replace the Sobol generator with the default RNG) so that `N[i]` is always filled by `my_rng.gauss()`; remove or comment out the `sobol_gen` lines and the `skip` argument. The rest of the routine Brownian Bridge construction, barrier logic, discounting and error computation remains unchanged. This gives an ordinary Monte Carlo price with identical time discretisation.

Listing C.1: QMC/MQMC function in Quantlab

```
out void price_barrier_option_qmc_BB(
number  rate,
number  vol,
number  time_to_maturity,
number  barrier_level,
number  strike,
integer n_steps, // must be 2^n (for midpoint path builder)
integer n_qsteps, //must be 2^m with m <= n  for RQMC
integer n_paths,
integer skip,        // Sobol skip
logical is_call,

// Existing out parameters:
out vector(number) exact_value_UO_BO,  // exact barrier price
out vector(number) mc_value_BO_BB,     // final MC price

// New out parameter: standard error
out vector(number) mc_std_error_BO_BB,
rng option(nullable) my_rng = null)
{
        number spot = 100;
        integer seed = millisecond(now());
        // track sums of payoffs
        number sum_payoffs    = 0;
        number sum_sq_payoffs = 0;

        vector(number) samples[n_paths];

        // Quasi-random draws for building Brownian paths
        sobol_gen s = new sobol_gen(n_qsteps, skip);
```

```
31
32          // For bridging barrier events
33          if(null(my_rng))
34                  my_rng = rng(seed);
35
36          number log_barrier_level = log(barrier_level);
37          number log_spot = log(spot);
38          number log_S_next;
39          number log_S_prev;
40
41  // 1) Quasi-random normals
42          vector(number) Zs[n_qsteps];
43          vector(number) N[n_steps];
44
45          for(integer path_i = 0; path_i < n_paths; path_i++)
46          {
47                  Zs = s.gauss();
48                  for(integer i = 0; i < n_steps; i++)
49                  N[i] = i < n_qsteps ? Zs[i] : my_rng.gauss();
50                  // 2) Brownian-bridge path W(t)
51                  vector(number) times, W;
52                  buildBrownianPathMidpointFree(
53                      n_steps,
54                      time_to_maturity,
55                      N,
56                      times,
57                      W);
58
59                  // 3) Evolve S(t) & check barrier
60                  logical knocked_out = false;
61                  log_S_prev = log_spot;
62
63                  for(integer i = 0; i < n_steps && !knocked_out; i
                        ↪ ++)
64                  {
65                          number dt = times[i+1] - times[i];
66                          number dw = W[i+1] - W[i];
67
68                          log_S_next = log_S_prev + ((rate - 0.5*vol*
                                ↪ vol)*dt + vol*dw);
69
70                          if(log_S_next > log_barrier_level)
71                          {
72                                  knocked_out = true;
73                                  break;
74                          }
75                          else if(log_S_prev < log_barrier_level &&
                                ↪ log_S_next < log_barrier_level)
76                          {
77                                  number numerator = (
                                        ↪ log_barrier_level - log_S_prev
                                        ↪ )*(log_barrier_level -
                                        ↪ log_S_next);
78                                  number denom     = (vol * vol) * dt
                                        ↪ ;
79                                  number log_p_exit    = (-2.0 * (
```

```
                                        ↪numerator / denom));
80                                  number test = my_rng.uniform();
81                                  if(test == 0 || log(test) <
                                        ↪log_p_exit)
82                                  {
83                                          knocked_out = true;
84                                          break;
85                                  }
86                          }
87
88                          log_S_prev = log_S_next;
89                  }
90
91                  // payoff if still alive
92                  number payoff = 0;
93                  if(!knocked_out)
94                  {
95                          payoff = is_call
96                                  ? max(exp(log_S_prev) -
                                        ↪strike, 0)
97                                  : max(strike - exp(
                                        ↪log_S_prev), 0);
98                  }
99
100                 samples[path_i] = payoff;
101
102         }
103
104         sum_payoffs = v_sum(samples);
105         sum_sq_payoffs = samples * samples;
106
107
108         // discount the average payoff
109         number mean_payoff = sum_payoffs / n_paths;
110         number var_payoff  = (sum_sq_payoffs / n_paths) - (
                ↪mean_payoff * mean_payoff);
111
112         number discounted_mean = exp(-rate * time_to_maturity)*
                ↪mean_payoff;
113         number std_dev         = sqrt(var_payoff)* exp(-rate*
                ↪time_to_maturity);
114         number std_err         = std_dev / sqrt(n_paths);
115         mc_value_BO_BB      = [discounted_mean];
116         mc_std_error_BO_BB  = [std_err];
117         // exact up-and-out barrier price
118         exact_value_UO_BO = [UO_Barrier(
119                 spot,
120                 time_to_maturity,
121                 strike,
122                 rate,
123                 vol,
124                 barrier_level)];
125 }
```

TRITA - SCI-GRU 2025:098
Stockholm, Sweden 2025

www.kth.se