Chapter 2

A Computational Study on Sobol' Sequences

Bahri Tokmak[1]

Ömür Ugur[2]

Abstract

This study presents a computational comparison between Quasi-Monte Carlo (QMC) methods based on Sobol' sequences and traditional Monte Carlo (MC) methods using the Mersenne Twister (MT) generator. While Sobol' sequences are widely recognized for outperforming MT in terms of convergence, our results reveal notable deficiencies when applied to high-dimensional Geometric Asian option pricing. To investigate this behavior, we conduct moment and correlation analyses, identifying a bias in the incremental construction of Sobol' paths—a bias that is absent in MT and can be alleviated through skipping initial points, scrambling, or Brownian Bridge (BB) techniques. All simulations are implemented in Python, with additional acceleration achieved through Graphics Processing Unit (GPU)-based parallel computing environments.

INTRODUCTION

The motivation for this work stems from the complexities of pricing exotic derivatives under models demanding numerous time steps, thereby creating highly dimensional Brownian motion trajectories. In financial engineering, it is often impossible to derive closed-form solutions for the valuation of financial products, especially those categorized as exotic options. As a result, numerical techniques—and Monte Carlo (MC) simulation methods in particular—play a critical role. As discussed extensively by Glasserman (Glasserman, 2014), the appeal of Monte Carlo methods lies in their general applicability, especially in cases where analytical solutions are infeasible or unavailable. In MC simulations, the proper use of random number

1 Middle East Technical University, Institute of Applied Mathematics, Scientific Computing, Turkey

2 Middle East Technical University, Institute of Applied Mathematics, Scientific Computing, Turkey

https://doi.org/10.58830/ozgur.pub862.c3485

generators is vital, as incorrectly generated random sequences can lead to severe mispricing of financial products, potentially resulting in significant financial losses for institutions and individuals involved in their trading.

Traditional Monte Carlo method relies on pseudo-random number generators (PRNGs), such as the widely used Mersenne Twister, to simulate stochastic processes. However, the convergence rate of classical MC is typically $\mathcal{O}\left(\mathrm{N}^{-1/2}\right)$, where N is the number of samples. This achieving high accuracy requires generating millions of paths, which can be computationally intensive and slow to converge (Glasserman, 2014 and Jäckel, 2002).

Quasi-Monte Carlo (QMC) methods aim to improve upon this by replacing pseudo-random numbers with deterministic low-discrepancy sequences, such as Sobol' or Halton sequences. These sequences are designed to fill the simulation space more evenly, reducing clustering and improving convergence to approximately $\mathcal{O}\left(\frac{(\log N)^d}{N}\right)$ as stated in (Jäckel, 2002). In particular, Sobol' sequences have gained significant attention due to their simplicity, extensibility in high dimensions, and well-understood mathematical properties.

Despite their theoretical advantages, QMC methods are sensitive to implementation details. As noted by Glasserman (Glasserman, 2004), implementation choices such as scrambling—to improve uniformity and mitigate artifacts—and the ordering of dimensions can significantly influence the accuracy and stability of QMC results. Additionally, techniques like Brownian bridge (BB) construction are often employed in QMC to reorder variance allocation in time-dependent simulations, thereby improving efficiency for path-dependent derivatives.

This study provides a practical and empirical evaluation of Sobol' sequences in high-dimensional Monte Carlo simulations. We benchmark the performance of Sobol implementation against the Mersenne Twister, across a set of controlled experiments that include:

- One- and multi-dimensional integration tests
- Geometric Asian option pricing under a geometric Brownian motion model
- Moment and correlation analysis among Sobol' dimensions
- Effects of scrambling, skipping initial points, and Brownian bridge
- Graphics Processing Unit (GPU)-accelerated simulations

Our main objective is to determine under what conditions QMC methods, particularly Sobol' sequences, truly outperform traditional pseudo-random simulations in practice, and to investigate the performance gains offered by GPU-based parallel computation in high-dimensional Monte Carlo simulations.

BACKGROUND AND RELATED WORK

Monte Carlo Methods

Monte Carlo methods are widely used to approximate expectations by drawing random samples from a given probability distribution (Glasserman, 2004). These methods operate by repeatedly sampling and evaluating a function of interest, thereby estimating integrals or probabilities numerically. As the number of samples increases, the estimate converges to the true value by virtue of the law of large numbers. Furthermore, the central limit theorem enables quantification of the uncertainty in the estimate by providing asymptotic confidence intervals and standard errors.

To illustrate the method, let us use the example from Section 1.1.2 of (Glasserman, 2004). Consider the pricing of a European call option using the Black Scholes framework. The payoff of the option depends on the terminal value of the underlying stock, which follows a lognormal distribution as a result of modeling the asset price dynamics with a Geometric Brownian Motion (GBM). By generating standard normal samples, we can simulate the terminal stock price and compute the discounted payoff. The process is summarized in Algorithm 1 (see Figure 1), which shows the basic steps for estimating the expected present value of the option using Monte Carlo simulation. In this algorithm, r denotes the risk-free interest rate, ó is the volatility of the underlying asset, S_0 is the initial asset price, K is the strike price, T is the time to maturity, and $Z_i \sim \mathcal{N}(0,1)$ represents independent standard normal variables.

Algorithm 1 Monte Carlo estimation of expected present value of a European call option

1: **for** $i \leftarrow 1$ to n **do**
2: Generate $Z_i \sim \mathcal{N}(0,1)$
3: $S_i(T) \leftarrow S_0 \cdot \exp\left(\left(r - \frac{1}{2}\sigma^2\right)T + \sigma\sqrt{T} \cdot Z_i\right)$
4: $C_i \leftarrow e^{-rT} \cdot \max(S_i(T) - K, 0)$
5: **end for**
6: Estimate: $\hat{C}_n = \frac{1}{n}\sum_{i=1}^{n} C_i$

Figure 1. Monte Carlo estimation of a European call option

In classical MC simulation, PRNGs such as the widely used Mersenne Twister are employed to sample independent and identically distributed random variables from target distributions. Although these methods exhibit a convergence rate of $\mathcal{O}\left(N^{-1/2}\right)$, their efficiency diminishes in problems that require high precision or suffer from the curse of dimensionality. See, for instance (Glasserman, 2004 and Jäckel, 2002).

To address these limitations, QMC methods utilize *low-discrepancy sequences (LDS)* to fill the integration domain more uniformly than pseudo-random samples. This deterministic structure can improve the convergence rate to $\mathcal{O}\left(\frac{(\log N)^d}{N}\right)$ for d-dimensional integrals under sufficient smoothness conditions (Glasserman, 2004, Jäckel, 2002 and Sobol' and Kucherenko, 2005). The discrepancy of a point set intuitively measures its deviation from uniformity, where lower discrepancy yields better space-filling.

Pseudo-Random Generators: Mersenne Twister

The Mersenne Twister is one of the most commonly used PRNGs due to its long period $2^{19937}-1$, fast generation speed, and guaranteed equidistribution in at least 623 dimensions (Jäckel, 2002). While its output is not truly random, it produces sequences that behave uncorrelated and independent for practical purposes, making it suitable for many high-dimensional Monte Carlo simulations.

In comparative studies, MT serves as a baseline for measuring QMC effectiveness, particularly in terms of convergence rate, variance, and dimensional correlation.

Low-Discrepancy Sequences: Sobol'

One of the most widely adopted low-discrepancy sequences in QMC methods is the Sobol' sequence, originally introduced in (Sobol', 1967) by I. M. Sobol' in 1967 and further developed in 1976. See (Sobol', 1976). These sequences are constructed using *direction numbers* and *Gray code* ordering to ensure uniform coverage over the unit hypercube $[0,1)^d$. Due to their simplicity of construction, scalability to high dimensions, and successful applications in finance, Sobol' sequences have become a standard choice in QMC simulations (Jäckel, 2002).

Despite their advantages, the effectiveness of Sobol' sequences in high-dimensional settings is not always guaranteed. High-dimensional implementations, such as BRODA's Sobol65536 generator as documented in (Broda, 2025), aim to scale QMC techniques to tens of thousands of

dimensions. However, empirical studies (Sobol', 1967 and Silva and Barbe, 2005) have observed issues such as dimensional correlation and loss of uniformity, which can negatively impact simulation accuracy. This is especially problematic in financial applications, where small deviations in distribution quality may lead to significant pricing and risk estimation errors.

Several practical enhancements have been proposed to improve the robustness of Sobol' sequences. Among these, three key factors are frequently highlighted:

- The use of *scrambling*, which introduces controlled randomness to eliminate structural artifacts and enable error estimation (Owen, 1998)
- Proper *dimensional ordering*, which assigns the most influential variables (if known) to the earliest dimensions, where Sobol' sequences typically exhibit better uniformity properties (Silva and Barbe, 2005)
- Careful treatment of the *initial points*, especially the very first point (typically $(0,0,\ldots,0)$ (Owen, 2021)

Scrambling was first introduced by Owen (Owen, 1998) to address the deterministic structure of low-discrepancy sequences, which limits error estimation through confidence intervals. In scrambling, the digits of the b-ary expansion of each point are randomly permuted in a recursive and structured manner. At the j-th digit, there are b^{j-1} partitions, each independently shuffled. This technique improves space-filling properties while retaining the low-discrepancy nature of the original sequence.

Another discussed topic in literature is the practice of skipping initial Sobol' points (also known as fast-forwarding), which involves discarding the first few values of the sequence—typically starting from the all-zero point $(0,0,\ldots,0)$. This is often motivated by the observation that early Sobol' points can lead to undesirable properties, particularly when passed through nonlinear transformations such as the inverse normal CDF $\Ö^{-1}$, to produce normally distributed variates. Under certain conditions, skipping has been shown to reduce numerical integration error, albeit without changing its asymptotic order (Radovic et al., 1996). However, the optimal number of points to skip is problem-dependent and cannot easily be determined. Moreover, as Owen (Owen, 2021) notes, indiscriminate skipping—especially when combined with scrambling—may disrupt the digital net structure of the sequence and lead to worse convergence behavior. As such,

the handling of initial points demands careful consideration and should ideally be validated within the specific simulation context.

A more algorithmic enhancement that complements Sobol' sequences is the Brownian bridge construction, a technique designed to improve the dimensional efficiency of path simulations. Brownian Bridge reorders the discretization of stochastic processes to concentrate variance in the early time steps. This reordering aligns well with Sobol' sequences, whose early dimensions are more uniformly distributed, allowing the most critical components of the process to benefit from the best uniformity. As a result, Brownian bridge improves both dimensional ordering and convergence properties (Glasserman, 2004, Kucherenko and Shah, 2007 and Bianchetti et al., 2015).

Given the high computational demands of large-scale QMC simulations, leveraging parallel computing architectures has become essential. In particular, GPUs offer a powerful platform for accelerating simulations due to their high degree of parallelism. Recent work by Bernemann et al (Bernemann et al., 2011) demonstrated the feasibility and effectiveness of GPU-accelerated Sobol-based simulations for pricing exotic derivatives and performing calibration tasks.

In this study, we build upon these methodological advances by evaluating GPU-based implementations of Sobol' and Mersenne Twister generators. Our focus is on understanding both the theoretical conditions and practical configurations under which Sobol' sequences, possibly enhanced with Brownian Bridge and scrambling, can outperform traditional pseudo-random approaches. The ultimate goal is to achieve high numerical accuracy while benefiting from the significant speedups offered by parallel hardware.

METHODOLOGY

This section outlines the computational setup, simulation framework, and evaluation metrics used in this study to compare Sobol-based QMC methods with classical MC methods using PRNGs, specifically the Mersenne Twister.

Tools and Computational Environment

All experiments are implemented in Python 3.12 and executed on a machine equipped with an NVIDIA GPU (CUDA enabled). The following libraries are used for simulation and numerical computation:

- *scipy*: To generate low-discrepancy Sobol' sequences, with support for scrambling and skipping.

- *numpy*: For numerical operations and random number generation using Mersenne Twister.
- *cupy*: To enable GPU-accelerated vectorized simulation paths.
- *torch*: Used specifically to generate Sobol' sequences directly on the GPU, as CuPy does not provide native support for GPU-based quasi-random sequence generation

The *Sobol* class in Scipy incorporates several important features for high-dimensional QMC simulations. It supports up to 21201 dimensions by utilizing precomputed direction numbers generated by Kuo et al. (Joe and Kuo, 2008). This capability is particularly useful in financial applications involving long time horizons, such as Asian options with daily monitoring over extended periods.

The implementation also includes a two-stage scrambling technique (Matousek, 1998) to enhance uniformity and reduce structural artifacts in the sequence. Such scrambling consists of:

- *Left Linear Matrix Scrambling (LMS)*, where direction numbers are transformed via a non-singular lower-triangular matrix to maintain low discrepancy while improving uniformity.
- *Digital Random Shift*, which applies a uniform digital shift across all dimensions to introduce randomness.

By combining these features—extended dimensional support and scrambling techniques—Scipy Sobol implementation is well-suited for high-dimensional simulations in financial engineering applications involving complex payoff structures.

Experimental Design

To provide a fair comparison between PRNG-based Monte Carlo and Sobol-based QMC, each experiment follows the same simulation logic, differing only in the source of randomness and variance reduction techniques. The following factors are controlled across experiments:

Sample size: Fixed to powers of two (e.g., 4096, $2^{19}, 2^{20}$) to suit Sobol's structure.

Dimensionality: Ranges from 1D to 21201D depending on the test, constrained by the maximum dimensionality supported by Python. In practical scenarios such as Asian option pricing, the number of dimensions corresponds to the number of discretization steps over time (e.g., 3650 for 10 years of daily steps).

Skip values: Various skip levels are tested (e.g., 1024, 2^{19}) to investigate convergence sensitivity.

Scrambling: Tests are conducted to assess the impact of randomized Sobol' sequences.

Brownian Bridge: Time steps are simulated using Brownian Bridge construction to reallocate variance into lower dimensions.

GPU usage: High-dimensional path simulations and payoff evaluations are accelerated using GPU parallelization to reduce computation time.

Evaluation Metrics

To assess simulation accuracy and efficiency, the following quantitative metrics are recorded, where $\hat{V}_i$ is the estimated value from trial i, and V_{true} is the known theoretical value (e.g., for Asian options), and N is the sample size (or, number of replications):

$$\text{Mean Absolute Error (MAE):} = \frac{1}{N}\sum_{i=1}^{N}\left|\hat{V}_i - V_{\text{true}}\right| \quad (1)$$

$$\text{Maximum Absolute Error (MaxAE)} = \max_{1\le i\le N}\left|\hat{V}_i - V_{\text{true}}\right| \quad (2)$$

$$\text{Mean Error (Bias)} = \frac{1}{N}\sum_{i=1}^{N}\left(\hat{V}_i - V_{\text{true}}\right) \quad (3)$$

The elapsed time is computed using Python's time.time() function before and after simulation blocks. We remark that in high-dimensional tests, correlation matrices and variance of intermediate quantities are also recorded to study internal simulation stability. By standardizing the above setup, we ensure reproducibility and comparability across different random number sources, variance reduction techniques, and hardware acceleration strategies.

INTEGRAL TEST

To systematically evaluate the numerical accuracy of QMC and traditional MC methods, we conduct a series of one-dimensional and multi-dimensional integration experiments.

One-Dimensional Integral Test

To assess the basic numerical behavior of QMC methods in low-dimensional settings, we begin by evaluating a classical family of one-dimensional integrals of the form

$$(n+1)\int_0^1 x^n\,dx = 1, \quad n = 1, 2, \ldots, 20 \quad (4)$$

for which the analytical value is known to be exactly one for all integer values of n. This benchmark serves as a reliable testbed for quantifying integration error and convergence characteristics of different sampling methods, including pseudo-random sequences and low-discrepancy sequences such as Sobol'.

Each integral is approximated numerically using 4096 sample points. The performance of both methods is compared by computing the absolute error between the estimated and exact values across the tested range of n. For each n, the maximum and mean absolute errors are recorded in Table 1.

Table 1. Absolute errors for one-dimensional monomial integrals using Sobol and MT

n	**Sobol Max**	**Sobol Mean**	**MT Max**	**MT Mean**
1	0.00018	0.00012	0.03549	0.00716
2	0.00028	0.00018	0.05918	0.01114
3	0.00037	0.00024	0.07578	0.01416
4	0.00046	0.00031	0.08617	0.01666
5	0.00055	0.00037	0.09518	0.01884
6	0.00064	0.00043	0.10459	0.02079
7	0.00073	0.00049	0.11263	0.02256
8	0.00083	0.00055	0.11959	0.02421
9	0.00092	0.00061	0.12588	0.02574
10	0.00101	0.00067	0.13414	0.02720
11	0.00110	0.00073	0.14182	0.02858
12	0.00119	0.00079	0.14896	0.02991
13	0.00128	0.00085	0.15560	0.03117
14	0.00138	0.00092	0.16178	0.03239
15	0.00147	0.00098	0.16757	0.03357
16	0.00156	0.00104	0.17320	0.03470
17	0.00165	0.00110	0.17849	0.03580
18	0.00174	0.00116	0.18347	0.03687
19	0.00183	0.00122	0.18818	0.03791
20	0.00193	0.00128	0.19262	0.03892

The results demonstrate that Sobol' sequences yield significantly lower integration error compared to Mersenne Twister across all values of n. In particular, the maximum and mean absolute errors associated with Sobol' are consistently an order of magnitude smaller than those of MT, especially for higher-degree monomials as clearly seen in Table 1. These findings reinforce the established theoretical understanding that low-discrepancy sequences exhibit superior performance in deterministic numerical integration, especially in low-dimensional settings.

Multi-Dimensional Integral Test

To assess the accuracy and robustness of QMC methods in higher dimensions, we conducted a series of experiments based on the family of multi-dimensional integrals given in Equation 5:

$$\int_0^1 \cdots \int_0^1 \prod_{i=j}^{j+n-1} \left(1 + c_i \cdot \left(x_i - 0.5\right)\right) dx_j \cdots dx_{j+n-1} = 1 \quad (5)$$

evaluated for various values of j, n, and constant coefficients c_i The integral is designed to emphasize the role of boundary sampling: the larger the values of c_i, the more weight is placed on the integrand near the edges of the $[0,1]^n$ hypercube. This makes it an ideal stress test for comparing Sobol' sequences to MT, especially in terms of their ability to adequately capture extreme regions of the domain.

We tested several values of n ranging from 3 to 21,201 dimensions. In the small and moderate dimensional cases ($n < 21201$), we employed a sliding window approach over a large Sobol' matrix of shape (4096×21201) : for each offset j, we extracted n consecutive dimensions, computed the numerical integral, and recorded the corresponding absolute error. In the full-dimensional case ($n = 21201$), a batch-based method is adopted. We performed 100 independent simulations of 4096 Sobol' samples, each preceded by a skip of $j \cdot 4096$, and reported the maximum and average absolute error across batches.

All Sobol' sequences are generated with scrambling disabled and appropriate skip values. Pseudo-random samples are initialized with batch-specific seeds to ensure independence. Each integrand is evaluated, having fixed $c_i = c$, as:

$$f(x) = \prod_{i=1}^{n} \left(1 + c \cdot \left(x_i - 0.5\right)\right) \quad (6)$$

The results are summarized in Table 2, which reports the observed maximum and mean absolute errors for both Sobol and Mersenne Twister methods across different configurations.

Table 2. Multi-dimensional integral errors for various (n, c) combinations

n and c	Sobol Max Error	Sobol Mean Error	MT Max Error	MT Mean Error
3 and 0.5	0.0157	0.00023	0.0153	0.0031
5 and 0.5	0.0192	0.00047	0.0218	0.0041
5 and 0.5 (skip 2^{19})	0.0192	0.00027	0.0218	0.0041
5 and 0.3	0.0068	0.00023	0.0132	0.0024
30 and 0.3	0.0258	0.00263	0.0334	0.0063
1000 and 0.01	0.0021	0.00073	0.0042	0.0014
21201 and 0.0002	0.000068	0.000004	0.0004	0.0001

These results clearly demonstrate the advantage of Sobol' sequences, particularly as dimensionality increases and the coefficient c decreases. In low-dimensional settings, both methods perform reasonably well, although Sobol still shows a lower mean absolute error. However, in high dimensions—especially with $n = 21201$ and $c = 0.0002$—Sobol significantly outperforms Mersenne Twister, with an error reduction factor exceeding 25 times.

An additional experiment applying a large skip of 2^{19} in the $n=5,\ c=0.5$ case also confirmed that skipping early Sobol' points leads to meaningful error reduction (from 0.00047 to 0.00027 in average error), without affecting the maximum deviation. See Table 2, thereby, we may claim that skipping can be beneficial for high-accuracy results.

Overall, these findings validate the superiority of Sobol' sequences equily well also in high-dimensional numerical integration.

GEOMETRIC ASIAN OPTION PRICING

To investigate the comparative performance of QMC methods and classical MC methods in a realistic financial context, we conducted a series of experiments pricing a geometric Asian call option under a Geometric Brownian Motion (GBM) framework: Black-Scholes setting.

The underlying asset price S_t is modeled according to the classical GBM process, defined by the stochastic differential equation:

$$dS_t = rS_t\,dt + \sigma S_t\,dW_t \quad (7)$$

where r denotes the constant risk-free rate, σ is the constant volatility of the asset, and W_t represents a standard Brownian motion. Discretization is performed using a log-Euler scheme:

$$\ln S_{t_i} = \ln S_{t_{i-1}} + \left(r - \frac{\sigma^2}{2} \right) \Delta t + \sigma \sqrt{\Delta t} \cdot z_i \quad (8)$$

where $z_i \sim \mathcal{N}(0,1)$ are independent standard normal variates generated via either pseudo-random or quasi-random sequences.

The geometric Asian call option payoff is based on the geometric mean of asset prices over discrete monitoring dates:

$$\text{Payoff} = \max\left(\text{GA} - K, 0\right), \quad \text{where} \quad \text{GA} = \left(\prod_{i=1}^{n} S_{t_i} \right)^{1/n} \quad (9)$$

with K denoting the strike price. In our experimental setup, we set $S_0 = 1$, $r = 0.05$, $\sigma = 0.4$ and $K = 1$. The maturity T corresponds to 21201 timesteps, equivalent to 10 monitoring points per day over approximately 5.8085 years:

$$t_i = \frac{i}{365 \times 10}, \quad i = 1, \ldots, 21201 \quad (10)$$

The theoretical value of the geometric Asian option is available in closed form. It is computed using an adjusted volatility and maturity:

$$\sigma_{\text{adj}} = \frac{\sigma}{\sqrt{3}}, \quad T_{\text{adj}} = \frac{T\left(r + \sigma^2 / 6\right)}{2}, \quad K_{\text{adj}} = K e^{T_{\text{adj}}} \quad (11)$$

yielding the option price:

$$\text{Price} = e^{-T_{\text{adj}}} \cdot \text{BlackScholes}\left(S_0, K_{\text{adj}}, r, \sigma_{\text{adj}}, T_{\text{adj}}\right) \quad (12)$$

where the Black-Scholes pricing formula for European call options (Black and Scholes, 1973) is employed:

$$\text{BlackScholes}\left(S_0, K, r, \sigma, T\right) = S_0 \Phi\left(d_1\right) - K e^{-rT} \Phi\left(d_2\right), \quad (13)$$

$$d_1 = \frac{\ln\left(S_0 / K\right) + \left(r + \frac{1}{2}\sigma^2 \right) T}{\sigma \sqrt{T}}$$

$$d_2 = d_1 - \sigma \sqrt{T}$$

Here, $\Phi(\cdot)$ denotes the cumulative distribution function (CDF) of the standard normal distribution.

For the option under consideration the theoretical formula returns 0.19715. Simulations are conducted using two approaches: Sobol' sequences, and Mersenne Twister based pseudo-random numbers. Each simulation consists of 16 independent trials, each using 4096 paths, leading to a total of 65536 simulated paths per method.

The Mersenne Twister results yields a mean estimated option price of 0.19575, compared to the theoretical price of 0.19715. The average absolute error across trials is 0.00597, with a maximum absolute error of 0.01702. The variance of the estimated prices is 0.00005398, corresponding to a standard deviation of 0.00735. These results indicate a relatively tight distribution of price estimates around the theoretical value.

In contrast, simulations using the Sobol' sequence produced a mean estimated option price of 0.23363, which deviates from the theoretical price of 0.19715 by an absolute difference of 0.03648. Despite using a large number of simulated paths (65536), this level of discrepancy is considered suboptimal, as such sample sizes are typically expected to yield tighter convergence. The average absolute error across trials is 0.04617, and the maximum absolute error reaches to 0.33548. The price estimates shows a variance of 0.00755007 and a standard deviation of 0.08689. Furthermore, the observed minimum and maximum prices (0.18357 and 0.53263, respectively) indicate a much wider dispersion compared to the Mersenne Twister results, reflecting greater instability and less reliable convergence behavior.

This discrepancy in performance prompted a deeper investigation into the underlying causes of the observed deficiencies in the Sobol' sequence simulations. To this end, the next section conducts a series of statistical analyses aimed at diagnosing potential issues inherent to the structure of Sobol-generated paths. In particular, we focus on two key diagnostic tools: moment analysis, to evaluate the marginal distributions of individual Sobol' dimensions, and correlation analysis, to detect any unexpected dependencies between dimensions that could impair the sequence's uniformity and effectiveness.

ROOT CAUSE ANALYSIS OF THE OBSERVED DISCREPANCIES

Moment Analysis of Sobol' Sequences

In order to better understand the statistical properties of Sobol' sequences in high-dimensional simulations, we perform a detailed investigation of the first four moments --- mean, variance, skewness, and kurtosis --- of the sequences. Two cases are analyzed: the raw Sobol' sequences on $[0,1)$, and their transformation into standard normal space via the inverse cumulative distribution function (ICDF), Φ^{-1}.

We generated samples at sizes 2^{12}, 2^{16}, 2^{20}, 2^{24} and 2^{28}, corresponding to 4096, 65536, 1048576, 16777216, and 268435456 samples, respectively.

For the raw Sobol' samples (i.e., before applying ICDF), the results are summarized below and depicted in Table 3.

- The sample mean converges to 0.5 as the number of samples increases, aligning with the expected value for a uniform distribution.
- The sample variance also converges to $\frac{1}{12} \approx 0.08333333$, the theoretical variance of the uniform distribution.
- The skewness remaines effectively zero across all sample sizes, indicating symmetric distributions.
- The kurtosis stabilizeds at 1.8, as theoretically expected for uniform random variables.

Table 3. Moments of Raw Sobol' Sequence ($[0,1)$)

n_{samples}	Mean	Variance	Skewness	Kurtosis
2^{12} (4096)	0.499969	0.08333329	1.06×10^{-10}	1.79999865
2^{16} (65536)	0.499992	0.08333333	6.29×10^{-12}	1.79999999
2^{20} (1048576)	0.499999	0.08333333	2.74×10^{-14}	1.8
2^{24} (16777216)	0.499999	0.08333333	1.08×10^{-16}	1.8
2^{28} (268435456)	0.5	0.08333333	-2.77×10^{-19}	1.8

After applying the inverse cumulative normal function Φ^{-1} to transform the Sobol' points to standard normal deviates, the moments shown in Table 4 are observed to behave as follows:

- The mean converges rapidly towards 0, as expected for a standard normal distribution.

- The variance converges to 1, as theoretically expected.
- The skewness remaines near 0, indicating symmetry of the resulting normal distributions.
- The kurtosis approaches 3, matching the kurtosis of a standard normal distribution.

Table 4. Moments of Transformed Sobol' Sequence $(\mathcal{N}(0,1))$

n_{samples}	Mean	Variance	Skewness	Kurtosis
2^{12} (4096)	-2.23×10^{-4}	0.99908158	−0.002273	2.97942869
2^{16} (65536)	-5.99×10^{-5}	0.99993473	−0.000829	2.99790646
2^{20} (1048576)	-4.45×10^{-6}	0.99999588	-8.43×10^{-5}	2.9998241
2^{24} (16777216)	-3.11×10^{-7}	0.99999974	-7.54×10^{-6}	2.99998616
2^{28} (268435456)	-2.13×10^{-8}	0.99999998	-6.30×10^{-7}	2.99999896

These empirical findings align with the theoretical properties established in the literature.

Correlation Analysis in Sobol' Sequences

It is well-known that Sobol' sequences can exhibit substantial correlations between different dimensions, meaning that the generated quasi-random numbers in different dimensions are correlated. In the context of this study, references to inter-dimensional correlation specifically refer to the correlation between the quasi-random numbers assigned to different dimensions. This phenomenon is already mentioned in the early works on low-discrepancy sequences (Sobol', 1967). In (Sobol' et al., 2012), it is noted that, for a particular implementation of Sobol' numbers, "a test done with 2500 dimensions showed that 2449 pairs of consecutive dimensions have correlation greater than 70% (in absolute value)."

In our study, with 21201 dimensions and 4096 samples, we perform a similar analysis. Among the $\frac{21201 \times 21200}{2} = 224,730,600$ possible distinct dimension pairs, we observe that 415,391 pairs exhibit a correlation greater than 0.1, corresponding to approximately 0.18% of all pairs. Furthermore, 91,668 pairs show correlations exceeding 0.5, and 91,631 pairs exceed 0.6, both representing approximately 0.04% of all possible pairs. When considering even stronger correlations, 91,493 pairs show correlations greater than 0.7, but this number dropped sharply to only 110 pairs when the threshold is raised to 0.8, accounting for a negligible fraction of

0.00005% and highlighting that while moderate correlations are relatively common, extremely strong correlations remain very rare. Specifically among consecutive dimensions, we identified 10 pairs with correlation above 0.7. This is in stark contrast to the findings of (Sobol' et al., 2011), where nearly all consecutive pairs in their test showed high correlation—highlighting the improved behavior of the *Scipy Sobol* implementation in high-dimensional settings.

This strong correlation behavior in Sobol' sequences contrasts sharply with the performance of pseudo-random generators such as Mersenne Twister. In a comparable experiment using Mersenne Twister with 4096 samples across 21201 dimensions, the maximum absolute correlation observed is only 0.091 (between dimensions 15689 and 15966).

Additionally, it is important to highlight that the magnitude of these correlations in Sobol' sequences diminishes significantly as the number of samples increases. For instance, considering the dimension pair (1229, 6014), the correlation is $\rho = 0.9375$ with 4096 samples but drops dramatically to $\rho = 0.0249$ when the number of samples are increased to 20000. This behavior is consistent with the theoretical expectations regarding the asymptotic behavior of Sobol' sequences.

Some representative scatter plots of correlated dimension pairs are shown in Figure 2. These visualizations illustrate how severe clustering can occur when the random variables associated with different dimensions are strongly correlated.

Following the application of the inverse standard normal transformation to the Sobol' samples, we performed a similar correlation analysis. Among the 224,730,600 possible distinct dimension pairs, 847,904 pairs exhibit correlations greater than 0.1, corresponding to approximately 0.38% of all pairs. Furthermore, 91,668 pairs show correlations exceeding 0.5, while 91,394 pairs exceed 0.6, both representing approximately 0.04% of all possible pairs and indicating that moderate levels of dependency remain relatively prevalent. However, as the correlation threshold is increased beyond 0.6, a sharp decline is observed: only 189 pairs exhibited correlations greater than 0.7, accounting for a minuscule proportion of approximately 0.00008% of all pairs. This sharp drop between 0.6 and 0.7 thresholds differs slightly from the pattern observed before the transformation, where a comparable decline is only observed between 0.7 and 0.8 thresholds. These findings suggest that the inverse transformation modifies the underlying dependency structure of Sobol' sequences, effectively reducing the prevalence

of very high correlations (greater than 0.7) and shifting dependencies into the correlation range (above 0.6).

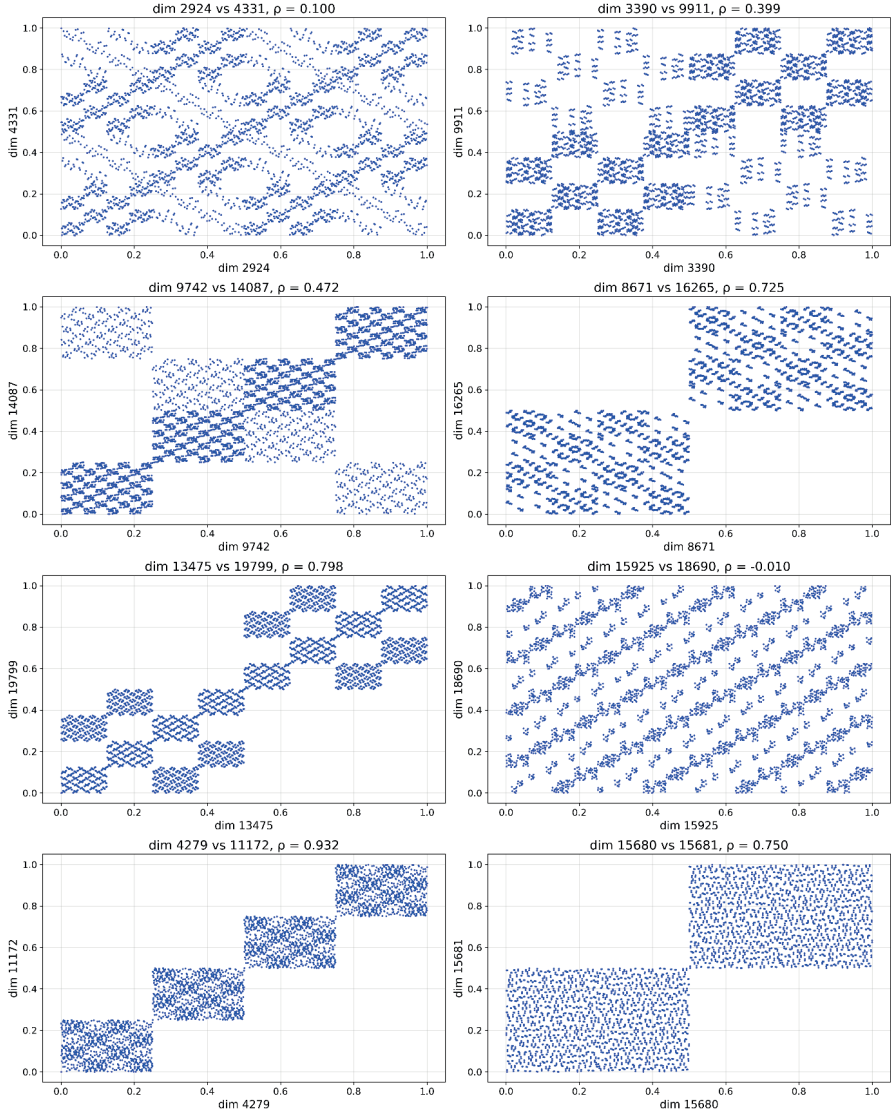

Figure 2. Scatter plots of selected dimension pairs from Sobol' sequence.

Variance Bias in Incremental Construction

In classical Monte Carlo simulation of Brownian motion, the standard discretization method constructs the terminal value W_T by sequentially summing small increments generated from independent Gaussian variables.

This method, commonly referred to as the incremental construction, approximates W_T as:

$$W_T^I = \sum_{i=1}^{N} \sqrt{dt}\, z_i \#(14)$$

where $z_i \sim \mathcal{N}(0,1)$ are independent standard Gaussian random variables, N is the number of time steps, and $dt = T/N$.

In an ideal Monte Carlo settings, if the z_i are truly independent, the following properties would hold:

$$E[z_i] = 0, \quad \forall i \#(15)$$

$$\mathrm{V}[z_i] = 1, \quad \forall i \#(16)$$

$$E[z_i z_j] = 0, \quad \text{for } i \neq j \#(17)$$

where $(E[\cdot])$ and $(V[\cdot])$ denote the expectation and variance operators, respectively. Consequently, the mean and variance of (W_T^I) satisfy:

$$\mathrm{E}[W_T^I] = 0 \#(18)$$

$$V[W_T^I] = T \#(19)$$

However, when quasi-random sequences like Sobol' numbers are used to generate z_i via the inverse normal CDF, strict independence between coordinates is no longer guaranteed. Particularly at small sample sizes or high dimensions, residual correlations between z_i and $z_j\,(i \neq j)c$ an introduce a bias into the variance of W_T^I.

To quantify this effect, assume that the empirical variance of each z_i is $v \lesssim 1$, and that the empirical correlation between different dimensions is $\tilde{\mathrm{n}}_{ij} = E[z_i z_j] - E[z_i]E[z_j]$. Then, the variance of W_T^I becomes:

$$V[W_T^I] = \sum_{i=1}^{N} dt\, V[z_i] + 2\sum_{i<j} dt\, \rho_{ij} \#(20)$$

$$= T\left(v + \frac{2}{N}\sum_{i<j}\rho_{ij}\right) \#(21)$$

We define the variance distortion term C induced by the correlations as:

$$C = \frac{2}{N}\sum_{i<j}\rho_{ij} \#(22)$$

such that the total variance is expressed as $T(v + C)$.

To complement the theoretical analysis, we computed the variance v, the term C caused by inter-dimensional correlations, and the total variance $v+C$ for both Sobol' and Mersenne Twister sequences across different dimensions. The sample size is fixed at 4096 simulations. The tested dimensionalities includes n_{dims} = 100, 500, 1000, 2000, and 21201, covering a wide range from low to extremely high dimensions. In Table 5, the empirical results are summarized. As observed, the Sobol' sequences exhibit significant deviations in the $v+C$ values, particularly for high dimensions such as $n_{\text{dims}} = 21201$, where $v+C$ exceeds 4.0. This reflects the impact of dimension-dependent correlations inherent to Sobol' sequences. In contrast, Mersenne Twister sequences show relatively stable behavior, with $v+C$ values remaining close to 1 even at high dimensions, confirming their low-correlation pseudo-random nature.

Table 5. Variance and variance distortion for Sobol' and Mersenne Twister sequences at $n_{\text{samples}} = 4096$

Method	n_{dims}	v	C	$v+C$
Sobol	100	0.999275	-0.044254	0.955021
Sobol	500	0.999275	-0.141961	0.857314
Sobol	1000	0.999278	-0.308172	0.691105
Sobol	2000	0.999277	-0.386930	0.612347
Sobol	21201	0.999282	3.061985	4.061268
MT	100	0.998997	-0.013125	0.985872
MT	500	0.998894	0.014969	1.013864
MT	1000	0.998440	-0.003114	0.995326
MT	2000	0.998587	-0.018400	0.980186
MT	21201	0.999428	0.001260	1.000688

Effect of Scrambling, Skipping, and Brownian Bridge on Variance Stability

In the previous sections, we observed that in the absence of any enhancements, Sobol' sequences exhibited significant deterioration in the total variance $v+C$ as the dimensionality increased. In this section, we explore how three different techniques — scrambling, skipping initial points, and Brownian Bridge construction — affect the variance stability.

Variance Stability under Scrambling

When scrambling is enabled, Sobol' sequences show a remarkable improvement in variance properties. As shown in Table 6, even for $n_{\text{dims}} = 21201$, the total variance $v+C$ is corrected to approximately 1.0025. Across all tested dimensions (n_{dims} = 100, 500, 1000, 2000, 21201), the deviations of $v+C$ from 1 remain within ± 0.006 margins, indicating stability.

Thus, the results observed in our tests validate the theoretical expectations: scrambling substantially improves the variance stability of Sobol' sequences, making them highly reliable even when applied to very high-dimensional integration problems.

Variance Stability under Initial Points Skipping

Applying a large skip of $2^{19} = 524,288$ points (without scrambling) results in a notable improvement in the stability of total variance estimates. For lower dimensions (n_{dims} = 100, 500), the total variance $v+C$ remains very close to the theoretical value of 1, with deviations on the order of 10^{-3} or less. Even in extremely high-dimensional settings, such as n_{dims} = 21201, the total variance remains stable around 0.9934 (see Table 6), representing a substantial improvement over the non-skipped case.

These results suggest that skipping a substantial number of initial Sobol' points can also substantially improve numerical behavior across a wide range of dimensions. In our tests, skipping is consistently observed to have a positive impact on stability, supporting the understanding that Sobol' sequences tend to perform better when an initial portion of the sequence is discarded.

Several studies, including Owen (Owen, 2021), have pointed out that skipping initial points in *scrambled* Sobol' sequences may disrupt the underlying randomized net structure, potentially degrading convergence. While our findings demonstrate the practical benefits of skipping in *non-scrambled* settings, further research is needed to better understand how skipping interacts with scrambling in different simulation contexts, particularly with respect to determining how many points should be skipped, which may vary significantly depending on the integrand, dimensionality, and target accuracy.

Variance Stability under Brownian Bridge Construction

Brownian bridge (BB) construction is a widely used technique to improve the efficiency of path generation in Monte Carlo and Quasi-Monte Carlo simulations. By carefully redistributing the variance contributions

across dimensions, BB can significantly stabilize the numerical behavior, particularly in high-dimensional settings.

Consider a standard Brownian motion process $W(t)$ over the interval $[0,T]$, with fixed endpoints:

$W(0)=0$

$W_T^{\mathrm{BB}}=\sqrt{T}z_1$, where $z_1 \sim \mathcal{N}(0,1)$

It follows that $E\left(W_T^{\mathrm{BB}}\right)=0$ and $V\left(W_T^{\mathrm{BB}}\right)=T$ since $E(z_1)=0$ and $V(z_1)=1$.

Intermediate values of the Brownian motion are not generated sequentially in time. Instead, the process recursively fills in midpoints of the largest remaining intervals. Suppose we already have values at two time points T_i and T_{i+1}, and we want to generate the value at a time $t \in (T_i, T_{i+1})$. This is done using:

$$W(t)=\frac{T_{i+1}-t}{T_{i+1}-T_i}W(T_i)+\frac{t-T_i}{T_{i+1}-T_i}W(T_{i+1})+\sqrt{\frac{(T_{i+1}-t)(t-T_i)}{T_{i+1}-T_i}}\cdot z_k \quad \#(23)$$

where each $z_k \sim \mathcal{N}(0,1)$ is an independent standard normal random variable.

The first two terms represent a linear interpolation, while the third term introduces stochasticity consistent with Brownian motion's properties. In theory, when ideal random numbers are used, the Brownian Bridge construction should maintain stability across dimensions without degradation as dimension increases. Minor deviations from the ideal variance are primarily attributed to residual correlations among quasi-random samples. That is, while Brownian bridge organizes the variance efficiently, quasi-random sequences like Sobol' are not perfectly independent across dimensions, and small correlations can slightly affect variance. See (Sobol' and Kucherenko, 2005, Kucherenko and Shah, 2007 and Bianchetti et al., 2015) for more information.

In our numerical experiments, we observe that when Brownian Bridge construction is applied to Sobol' sequences, the total variance $v+C$ remains very close to 1 across all dimensions (see Table 6). Even at $n_{\mathrm{dims}}=21201$, the total variance is approximately 0.9989, reflecting agreement with theoretical expectations. These results validate that Brownian Bridge dramatically improves variance stability in high-dimensional settings, and small observed deviations are consistent with the minor correlation effects inherent to quasi-random sequences rather than flaws in the Brownian Bridge algorithm itself.

Table 6. Summary of Total Variance $v+C$ Under Different Enhancements

Method	Scrambling Enabled			
	Dimensions	Variance	Distortion	Total
Sobol	100	1.000169	0.002468	1.002636
Sobol	500	0.999994	-0.015104	0.984890
Sobol	1000	0.999974	0.006229	1.006202
Sobol	2000	1.000021	0.003872	1.003893
Sobol	21201	0.999992	0.002492	1.002484

Table 6. continued

Method	Skip = 2^{19} (524288 points)			
	Dimensions	Variance	Dimensions	Total
Sobol	100	1.000097	0.001626	1.001723
Sobol	500	1.000026	-0.000100	0.999926
Sobol	1000	1.000002	-0.004902	0.995100
Sobol	2000	1.000004	-0.014498	0.985507
Sobol	21201	0.999997	-0.006642	0.993355
Method	**Brownian Bridge Enabled**			
	Dimensions	**Variance**	**Distortion**	**Total**
Sobol	100	0.999275	-0.000388	0.998886
Sobol	500	0.999275	-0.000234	0.999040
Sobol	1000	0.999278	0.000012	0.999290
Sobol	2000	0.999277	-0.000324	0.998953
Sobol	21201	0.999282	-0.000391	0.998892

EFFECT OF SCRAMBLING, SKIPPING, AND BROWNIAN BRIDGE ON ASIAN OPTION PRICING

To evaluate the practical impact of various Sobol' sequence enhancement techniques on financial simulations, we conducted a series of controlled experiments focusing on the pricing of a Geometric Asian option. The theoretical price, previously established in Section "geometric asian option pricing", serves as a benchmark for assessing pricing accuracy.

In this study, we specifically tested the effects of scrambling, initial skipping, and Brownian bridge construction. Four different Sobol configurations are examined:

- *Baseline*: No scrambling, no Brownian Bridge.
- *Scrambled Sobol*: Scrambling enabled, no Brownian Bridge.
- *Brownian Bridge*: No scrambling, Brownian Bridge enabled.
- *Large Skip Only*: Skip = 2^{19} (524,288 points), No scrambling, no Brownian Bridge.

Each configuration, except the initial skip case, is evaluated over 16 independent trials with 4096 paths per trial to ensure statistical consistency. The initial skip case configuration used a single batch of 4096 paths without repetition across trials.

The results, summarized in Table 7, reveal clear and consistent trends regarding the effectiveness of these enhancement techniques. Additionally, results for the standard Mersenne Twister simulation have been included in the table to provide reference for comparison with theoretical MC.

The baseline configuration, without scrambling or Brownian bridge, performs poorly: the mean estimated price is 0.23363, deviating significantly from the theoretical value. The maximum absolute error reaches to 0.33548, and the variance across trials is as high as 0.00755. These large errors are primarily attributed to structural artifacts and correlations inherent in the raw Sobol' sequence.

Enabling scrambling leads to a substantial improvement. The mean price converges to 0.19775, with a maximum error of only 0.00908, and The variance is reduced significantly, reaching approximately to 1.542×10^{-5} . This supports theoretical findings that scrambling improves uniformity, reduces bias, and allows for effective variance estimation.

Applying Brownian bridge construction without scrambling further enhances performance. The mean price achieved is 0.19711, with a maximum absolute error of just 0.00121, and the variance is reduced to 2.4×10^{-7} . This dramatic variance stabilization is consistent with the theoretical expectation that Brownian bridge reallocates variance contributions, improving the convergence behavior, particularly in high-dimensional settings.

Finally, applying a large skip of 2^{19} points—without scrambling or Brownian bridge—also produces notably accurate results. The mean price is 0.19608, and the maximum absolute error is just 0.00107, indicating a meaningful reduction in simulation bias. While skipping alone may not match the variance stabilization achieved by Brownian bridge, it remains an effective and simple strategy for improving coverage and reducing structural artifacts in Sobol'-based simulations.

Table 7. Comparison of Different Configurations in Geometric Asian Option Pricing

Configuration	Mean Price	Avg Error	Max Error	Variance
MT	0.19575	0.00597	0.01702	0.00005398
No Scramble, No BB	0.23363	0.04617	0.33548	0.00755007
Scramble = True, No BB	0.19775	0.00319	0.00908	0.00001542
BB = True, No Scramble	0.19711	0.00041	0.00121	0.00000024
Skip = 2^{19} (1 batch)	0.19608	0.00107	0.00107	N/A

In summary, our experiments demonstrate that proper configuration of Sobol' sequences is crucial for achieving high precision in QMC simulations in finance. Scrambling significantly reduces bias and variance, while Brownian Bridge construction further stabilizes variance by optimizing the allocation of variability across dimensions. Initial skipping alone can offer measurable improvements, but the best results are achieved when Brownian Bridge technique are employed. These findings reinforce the necessity of combining enhancement strategies to fully exploit the potential of quasi-Monte Carlo methods in high-dimensional option pricing problems.

GPU-ACCELERATED SIMULATIONS

The use of GPUs in computational finance has been extensively explored in the literatüre (Dempster et al., 2018). The inherently parallel structure of Monte Carlo simulations for path-dependent option pricing makes them well-suited for GPU-enabled parallel computing frameworks like CUDA. In our study, we harness the parallel processing power of GPUs to enhance the speed and efficiency of option pricing computations.

In our implementation, we primarily utilize the CuPy library to perform all array operations, random number transformations, and Monte Carlo path simulations on the GPU. CuPy provides a highly efficient, NumPy-compatible interface that allows straightforward migration of CPU-based codes to CUDA-enabled devices with minimal adjustments. PyTorch is employed exclusively for generating Sobol' sequences directly on the GPU, as CuPy currently lacks a native GPU-based Sobol generator. By using PyTorch's *Sobol Engine* for quasi-random number generation and relying on CuPy for the remaining computational tasks, we combine the strengths of both libraries to maximize performance and maintain numerical accuracy in high-dimensional Monte Carlo simulations.

This section presents a comparative evaluation of GPU-accelerated Monte Carlo and Quasi-Monte Carlo simulations in the context of pricing Geometric

Asian options (BS framework). To assess the impact of GPU acceleration, we conducted a series of controlled experiments focusing on runtime performance. Specifically, we benchmarked CPU-based versus GPU-based implementations across three scenarios: Mersenne Twister random number generation, Sobol' sequence generation, and Sobol' sequence generation combined with Brownian Bridge construction. These experiments are designed to isolate and quantify the computational advantages offered by GPU parallelization while keeping the pricing methodology and simulation parameters consistent. In all cases, CPU implementations are executed serially and serve as a baseline for evaluating the speedup and efficiency improvements achieved through GPU parallelization.

Algorithm 2 (see Figure 3) models the core simulation loop for Monte Carlo and Quasi-Monte Carlo simulations, where random numbers are either sampled from a standard normal distribution (for Mersenne Twister) or generated via Sobol' sequences followed by an inverse transformation. Each path is constructed through cumulative summation of the simulated increments and subsequently used to compute the option payoff. Algorithm 3 (see Figure 4) specifically describes the construction of Brownian Bridge increments using Sobol' sequences, to allocate early Sobol' dimensions to the most critical parts of the simulated Brownian motion path.

Algorithm 2 Per-thread vectorized path simulation using either Mersenne Twister or Sobol' sequences

$dt \leftarrow T/N$
$drift \leftarrow (r - 0.5\sigma^2) \cdot dt$
$vol \leftarrow \sigma \cdot \sqrt{dt}$
if Generator = Mersenne Twister **then**
    $Z \leftarrow$ StandardNormal[PATHS][N] ▷ Generated using `cupy.random`
else if Generator = Sobol **then**
    $U \leftarrow$ Sobol[PATHS][N] ▷ From `torch.quasirandom.SobolEngine`
    $Z \leftarrow \sqrt{2} \cdot \texttt{erfinv}(2U - 1)$ ▷ Inverse CDF transform
end if
$increments \leftarrow drift + vol \cdot Z$ ▷ Element-wise vectorized operation
$log\_paths \leftarrow \texttt{cumsum}(increments, \text{axis} = 1)$ ▷ Sequential in time, parallel across paths
$log\_paths \leftarrow \texttt{concat}([0], log\_paths)$ ▷ Insert $W_0 = 0$
$S \leftarrow S_0 \cdot \exp(log\_paths)$ ▷ Path prices
$GA \leftarrow \exp\left(\frac{1}{N}\sum_{i=1}^{N} \log S_i\right)$ ▷ Geometric mean per path
$payoff \leftarrow \max(GA - K, 0)$ ▷ Vectorized payoff calculation
$price \leftarrow \exp(-rT) \cdot \texttt{mean}(payoff)$

Figure 3. Illustrative figure of per-thread vectorized path simulation workflow using Mersenne Twister or Sobol' sequences

Algorithm 3 Brownian Bridge construction using Sobol-generated normal deviates

$U \leftarrow$ Sobol[PATHS][N] ▷ Sobol samples from `torch.quasirandom.SobolEngine`
$Z \leftarrow \sqrt{2} \cdot \texttt{erfinv}(2U - 1)$ ▷ Inverse CDF transform
Initialize $W_t[n] \leftarrow \sqrt{T} \cdot Z[t][0]$ ▷ Set terminal value
Build midpoint schedule using recursive bisection
$cur\_z \leftarrow 1$
for each midpoint (i, l, r) **do**
    $\Delta \leftarrow t_r - t_l$
    $a \leftarrow \frac{(t_r - t_i) \cdot W_t[l] + (t_i - t_l) \cdot W_t[r]}{\Delta}$
    $b \leftarrow \sqrt{\frac{(t_i - t_l)(t_r - t_i)}{\Delta}}$
    $W_t[i] \leftarrow a + b \cdot Z[t][cur\_z]$
    $cur\_z \leftarrow cur\_z + 1$
end for
$dw[t][i] \leftarrow (W_t[i+1] - W_t[i]) \cdot \sqrt{N/T}$ ▷ Construct Brownian increments

Figure 4. Illustration of Brownian Bridge path generation based on Sobol-normal samples

We also remark that Table 8 summarizes the execution times recorded for each simulation setup under a single batch execution.

Table 8. Execution time comparison for MC and QMC simulations (in seconds)

Method	Implementation	Time (s)
MT	CPU	3.148
MT	GPU	*0.095*
Sobol	CPU	4.889
Sobol	GPU	*0.570*
Sobol + BB	CPU	71.266
Sobol + BB	GPU	*5.788*
Sobol + Scramble	CPU	5.582
Sobol + Scramble	GPU	*1.085*
Sobol + (Skip = 2^{19})	CPU	18.108
Sobol + (Skip = 2^{19})	GPU	*13.552*

The results in Table 8 highlight the considerable computational advantages offered by GPU acceleration across various simulation methods. Among these, the most significant improvement is observed in standard Monte Carlo simulations using Mersenne Twister, where the GPU implementation completes the task in just 0.095 seconds, compared to 3.148 seconds on the CPU—a 33-fold speedup. This dramatic reduction demonstrates the efficiency of GPU-based parallel random number generation for large-scale simulations.

We next examine the performance of quasi-Monte Carlo simulations using Sobol' sequences. The GPU implementation achieves a runtime of 0.57 seconds, offering an 8.5-fold speedup relative to its CPU counterpart. This improvement underscores the effectiveness of GPU acceleration even when using low-discrepancy sequences, which are traditionally more structured and less amenable to parallelism than pseudo-random number generators. In the scrambled Sobol case, GPU runtime increases modestly to 1.085 seconds due to the additional computational cost of digital scrambling, yet it remains significantly faster than CPU execution.

Brownian Bridge construction, when applied alongside Sobol' sequences, introduces additional computational overhead due to its recursive midpoint structure. Nonetheless, the GPU implementation reduces runtime from 71.266 seconds on the CPU to 5.788 seconds, achieving a 12-fold acceleration. Despite this gain, BB simulations remain more time-consuming overall, as the recursive dependencies inherently limit parallelism on GPU architectures.

A more detailed breakdown of execution times reveals the primary computational bottlenecks in each configuration. For Sobol' simulations on GPU, approximately 86% of the total time is spent on generating quasi-random numbers, while the remaining time is used for path construction. In the scrambled Sobol case, the extra 0.515 seconds of overhead stems from scrambling operations. In contrast, applying a large skip (e.g., 2^{19}) leads to a total runtime of 13.552 seconds on GPU, indicating that while skipping improves sequence quality, it is computationally inefficient in practice.

In Brownian Bridge simulations, the performance bottleneck shifts away from number generation. Approximately 90% of the GPU runtime is spent on recursive midpoint interpolation, while only about 8% is used for generating the Sobol' sequence. This shift clearly shows that the recursive structure of BB—not the sampling method—is the dominant contributor to total execution time in this configuration.

These observations suggest that the combination of CuPy and PyTorch---both high-level GPU libraries---offers an effective and practical solution for accelerating Monte Carlo and Quasi-Monte Carlo simulations without requiring low-level custom CUDA kernel programming. High-level libraries like CuPy and PyTorch handle kernel generation and GPU memory management automatically, whereas low-level CUDA programming requires manually writing and optimizing custom kernels. Using such high-level libraries allows for quick implementation while maintaining sufficient

computational performance for pricing single exotic options, such as the Geometric Asian option considered in this study.

However, it is important to note that this analysis focuses on a single product type. In scenarios where portfolios of multiple exotic options are to be priced, or risk metrics such as Value-at-Risk (VaR) and Conditional Value-at-Risk (CVaR) are to be computed, the overall dimensionality (i.e., the total number of time steps × assets) —and thus the computational complexity— can become extremely large. In such cases, relying solely on high-level libraries may no longer be sufficient, and more optimized implementations involving explicit kernel configurations could be necessary to fully exploit the available GPU resources.

Lastly, the remarkable speedup observed with GPU-accelerated Mersenne Twister simulations suggests that further studies combining Mersenne Twister random number generation with advanced variance reduction techniques under full GPU parallelization could make valuable contributions to the computational finance literature. Moreover, exploring algorithmic modifications to the Brownian Bridge construction that improve its compatibility with parallel architectures presents another promising direction for enhancing the efficiency of quasi-Monte Carlo methods.

CONCLUSION

This study provides a comprehensive computational analysis of Quasi-Monte Carlo methods using Sobol' sequences in comparison to traditional Monte Carlo simulations based on the Mersenne Twister generator. While Sobol' sequences are theoretically known to offer superior convergence rates, our investigation reveals notable challenges when applying them to high-dimensional problems—arising, for instance, in financial simulations such as Geometric Asian option pricing.

The theoretical background and related literature highlight the known strengths of Sobol' sequences, but also hint at their sensitivity to dimension ordering and structural artifacts. Through one- and multi-dimensional integral tests, we validate the accuracy and convergence behavior in integral calculations using Sobol' sequences. However, when transitioning to the high-dimensional setting of Asian option pricing, significant deviations are observed, especially in baseline Sobol configurations.

To investigate the source of the observed discrepancies, we conduct detailed moment and correlation analyses; they reveal a persistent bias inherent in the incremental construction of Sobol' sequences. This bias is found to be effectively mitigated through various enhancement

techniques, including scrambling, initial skipping, and Brownian Bridge construction. Among these, Brownian bridge yields the most accurate option price estimates, while scrambling offers both improved accuracy and straightforward parallel implementation. Although initial skipping alone provides noticeable improvements, its overall effectiveness is more limited relative to Brownian Bridge technique.

Finally, we implement GPU-accelerated versions of all simulation methods, achieving substantial speedups across all configurations. The most striking gains are observed in Mersenne Twister-based simulations, where GPU parallelism yields a 33-fold runtime reduction of the base case. Sobol with Brownian Bridge simulations also benefits significantly from GPU acceleration, though to a lesser extent due to algorithmic limitations in parallelizing recursive path construction.

Overall, our findings reinforce the importance of proper configuration when applying QMC methods to high-dimensional problems. Moreover, they highlight the combined value of theoretical insight, algorithmic enhancement, and hardware-level acceleration in achieving both numerical precision and computational efficiency in modern financial simulations.

FUTURE WORK

The findings of this study suggest several directions for future research that could further enhance the efficiency, accuracy, and applicability of GPU-accelerated Monte Carlo and Quasi-Monte Carlo methods in computational finance.

First, the remarkable performance gains achieved through GPU-accelerated Mersenne Twister (MT) simulations indicate a valuable opportunity for further enhancement. In particular, the use of high-level GPU libraries such as CuPy enables the realization of substantial speedups without requiring low-level CUDA programming, making efficient pseudo-random number generation easily accessible within Python environments. While MT-based simulations offer excellent computational speed due to their lightweight nature, they are not inherently variance-reducing. Therefore, integrating MT with advanced variance reduction techniques—such as control variates, stratification, or antithetic sampling—under full GPU parallelization could substantially improve simulation accuracy without sacrificing computational efficiency. This combination may serve as a practical and scalable alternative to Quasi-Monte Carlo methods in applications.

Second, although the Brownian Bridge construction is a well-established variance reduction technique, its recursive nature limits full parallelization,

especially on GPU architectures. One fruitful direction for future work would be to investigate alternative algorithmic structures or approximations that retain the variance allocation benefits of Brownian Bridge while reducing memory bottlenecks or enabling greater parallel throughput. Additionally, it would be worth exploring whether custom low-level CUDA kernel implementations—specifically designed to optimize memory access and thread scheduling—could further enhance the performance of Brownian Bridge-based simulations beyond what is achievable with high-level GPU libraries alone.

Third, some studies, particularly in the context of *scrambled* Sobol' sequences, caution that skipping initial points may interfere with the randomized net structure and degrade convergence (Owen, 2021). Nonetheless, other works such as (Radović et al., 1996) report reduced integration error when early low-quality points are avoided. Our results support this view in the *non-scrambled* case: skipping a large number of initial points (e.g., 2^{19}) improved accuracy in high-dimensional simulations without Brownian Bridge. In addition to the interaction between skipping and scrambling, the question of how many points should be skipped remains problem-dependent, underscoring the need for more systematic analysis across different integrands and dimensionalities to better understand when skipping improves or degrades performance.

References

Bernemann A., Schreyer R., Spanderen K., 2011. Accelerating Exotic Option Pricing and Model Calibration Using GPUs, Available at SSRN 1753596.

Bianchetti M., Kucherenko S., Scoleri S., 2015. Pricing and Risk Management with High-Dimensional Quasi Monte Carlo and Global Sensitivity Analysis, Wilmott, 2015, (78): 46-70.

Black F., Scholes M., 1973. The Pricing of Options and Corporate Liabilitie, Journal of Political Economy, 81 (3): 637-654.

Broda, 2025. Access address: https://www.broda.co.uk/index.html; Date of Access: 30.05.2025

Dempster M.A.H., Kanniainen J., Keane J., Vynckier E., 2018, High-Performance Computing in Finance: Problems, Methods, and Solutions. 1st ed., CRC Press, Taylor & Francis Group, UK.

Glasserman P., 2003, Monte Carlo Methods in Financial Engineering. 1st ed., Springer, USA

Jäckel P., 2002, Monte Carlo Methods in Finance. 1st ed., John Wiley & Sons, UK

Joe S., Kuo F.Y., 2008. Constructing Sobol' Sequences with Better Two-Dimensional Projections, SIAM Journal on Scientific Computing, 30 (5): 2635-2654

Kucherenko S., Shah N., 2007. The Importance of being Global.Application of Global Sensitivity Analysis in Monte Carlo option Pricing, Wilmott, volume (issue): 82-91.

Matoušek J., 1998. On the L2-discrepancy for anchored boxes, Journal of Complexity, 14 (4): 527-556

Owen A.B., 1998. Scrambling Sobol' and Niederreiter–Xing Points, Journal of Complexity, 14 (4): 466-489

Owen A.B., 2021. On dropping the first Sobol' point. Access address: https://arxiv.org/abs/2008.08051; Date of Access: 30.05.2025.

Radović I., Sobol' I.M., Tichy R.F., 1996. Quasi-Monte Carlo Methods for Numerical Integration: Comparison of Different Low Discrepancy Sequences. Monte Carlo Methods and Applications, 2 (1): 1-14.

Silva M.E., Barbe T., 2005. Quasi-Monte Carlo in finance: extending for problems of high effective dimension, Economia Aplicada, 9 (4): 577-594.

Sobol' I.M., 1967. On the distribution of points in a cube and the approximate evaluation of integrals, USSR Computational Mathematics and Mathematical Physics, 7 (4): 86-112.

Sobol' I.M., 1976. Uniformly distributed sequences with an additional uniform property. USSR Computational Mathematics and Mathematical Physics, 16 (5): 236-242.

Sobol' I.M., Kucherenko S., 2005. On global sensitivity analysis of quasi-Monte Carlo algorithms, Monte Carlo Methods and Applications, 11 (1): 83-92.

Sobol' I.M., Asotsky D., Kreinin A., Kucherenko S., 2012. Construction and Comparison of High-Dimensional Sobol' Generators, Wilmott, 2011 (56): 64-79.

Appendix

GPU and CUDA specifications

CUDA toolkit version 12.6 was used for all of the simulations. The online documentation for this version is available at (https://docs.nvidia.com/cuda/archive/12.6.0/index.html). All simulations were conducted on a single NVIDIA GeForce RTX 4070 Laptop GPU. The specifications of the device are summarized in Table 9.

Table 9. Hardware specifications of the GPU used in simulations (NVIDIA GeForce RTX 4070)

Title	Title
Device Name	NVIDIA GeForce RTX 4070 Laptop GPU
CUDA Driver / Runtime Version	12.6 / 12.6
Compute Capability	8.9
CUDA Cores	4608
Global Memory	8 GB (8585 MB)
Shared Memory per Block	49 KB
Constant Memory	64 KB
L2 Cache Size	32 MB
Memory Clock Rate	8001 MHz
Memory Bus Width	128-bit
GPU Max Clock Rate	2175 MHz
Maximum Texture Dimension (1D)	131072
Maximum Texture Dimension (2D)	(131072, 65536)
Maximum Texture Dimension (3D)	(16384, 16384, 16384)
Maximum Threads per Multiprocessor	1536
Maximum Threads per Block	1024
Max Block Dimensions (x, y, z)	(1024, 1024, 64)
Max Grid Dimensions (x, y, z)	(2147483647, 65535, 65535)
Warp Size	32
Support for Concurrent Copy and Execution	Yes
Unified Memory (UVA) Support	Yes
ECC Support	No

Python Environment and Library Versions

All simulations were implemented using Python 3.12.4, supported by a set of high-performance numerical and GPU-accelerated libraries. The core packages and their versions are summarized in Table 10.

Table 10. Python Environment and Library Versions

Library	Version
Python	3.12.4
NumPy	1.26.4
SciPy	1.13.1
CuPy	13.3.0
PyTorch	2.5.1

Acknowledgment

This research was conducted as part of the TÜBİTAK 1001 project titled "Optimizing Conditional Value-at-Risk in Complex Portfolios through Parallel Computing Strategies" (Project No: 124F138), supported by the Scientific and Technological Research Council of Turkey (TÜBİTAK). The authors gratefully acknowledge TÜBİTAK for its financial support.

Conflict of Interest

The authors have declared that there is no conflict of interest.

Author Contributions

Both authors contributed equally to this work.