Variance Reduction in Monte Carlo Simulations for Financial Engineering

Camellini Francesco, Kim Youseung, Parietti Alessandro

November 2025

Contents

1 Introduction

1.1 Motivation

Monte Carlo (MC) simulation is one of the most fundamental numerical tools in financial engineering. It is widely used for pricing derivatives whose payoffs are non-linear, path-dependent, or analytically intractable. Although MC simulation is conceptually simple and universally applicable, its major drawback is the slow convergence rate, which is proportional to $\frac{1}{\sqrt{N}}$, where N is the number of simulated paths. In practical pricing applications—especially when evaluating exotic or rare-event-driven derivatives such as barrier or deep out-of-the-money options—achieving acceptable accuracy may require millions of simulations, leading to significant computational cost.

To overcome this issue, Variance Reduction Methods (VRMs) play a crucial role. These methods aim to produce estimators with the same expected value but smaller variance, thus achieving tighter confidence intervals without increasing the number of simulations.

This study provides a concise comparison of three variance reduction methods—Antithetic Variates, Control Variates, and Importance Sampling. By evaluating these methods across European, Asian, and Barrier options, the analysis illustrates how the structure of a payoff determines the achievable reduction in variance. The report also extends the classical Importance Sampling approach and implements a two-phase exponential tilting scheme for the Down-and-In option, offering a practical demonstration of how an appropriately chosen change of measure can stabilize rare-event simulations. Together, these results clarify the numerical gains offered by each method and the conditions under which they are effective, providing a guide for applying variance reduction in derivative pricing.

1.2 Problem Definition

The objective of this project is to evaluate and compare the effectiveness of three widely used variance reduction techniques: Antithetic Variates (AV), Control Variates (CV), and Importance Sampling (IS). Our analysis considers multiple derivative types with distinct payoff structures.

- European Call - path-independent, convex payoff
- Asian Call - path-dependent averaging payoff
- Barrier Call - path-dependent (event–driven) payoff that requires barrier activation

These options were chosen because they differ in complexity, smoothness, and distributional characteristics. The differences allow us to analyze how each VRM interacts with the underlying payoff structure. The central questions are the following:

1. Which variance reduction methods work best for which types of options?
2. Why do certain methods succeed or fail depending on payoff structures?
3. How much computational efficiency does each method provide?

Through theoretical derivations and extensive numerical experiments, we aim to provide a systematic comparison across methods and option types.

2 Mathematical Preliminaries

In the preliminaries, this report establishes the mathematical foundation used throughout the analysis. We first review the Monte Carlo estimator and its statistical properties, then introduce the Black–Scholes dynamics under the risk-neutral measure. Finally, we summarize the payoff definitions of the option contracts considered in this study. These elements collectively form the baseline simulation framework upon which all variance reduction techniques are applied.

2.1 Monte Carlo Estimator

Let X be a random variable with known distribution, and let $f : \mathbb{R} \to \mathbb{R}$ denote a measurable payoff function. The goal of Monte Carlo simulation is to approximate the expectation

$$\alpha = \mathbb{E}[f(X)].$$

Given independent samples $X_1, \ldots, X_n \overset{iid}{\sim} X$, the standard estimator is

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

Two fundamental probabilistic results justify the use of $\hat{\alpha}_n$:

Strong Law of Large Numbers (LLN).

$$\hat{\alpha}_n \xrightarrow{a.s.} \alpha.$$

LLN ensures the consistency of the estimator.

Central Limit Theorem (CLT). If $\sigma^2 = \mathrm{Var}(f(X)) < \infty$, then

$$\sqrt{n}\,(\hat{\alpha}_n - \alpha) \xrightarrow{d} N(0, \sigma^2),$$

which implies the asymptotic 95% confidence interval

$$\hat{\alpha}_n \pm 1.96 \frac{\hat{\sigma}_n}{\sqrt{n}},$$

where the sample variance

$$\hat{\sigma}_n^2 = \frac{1}{n-1} \sum_{i=1}^{n} (f(X_i) - \hat{\alpha}_n)^2$$

provides a consistent estimate of σ^2. The convergence rate of standard Monte Carlo is limited to $O(n^{-1/2})$, which motivates the use of variance reduction techniques.

2.2 Black-Scholes Model

Throughout the report, all option prices are computed under the risk-neutral measure $\mathbb{Q}$. The underlying asset price S_t follows a Geometric Brownian Motion (GBM):

$$dS_t = rS_t\,dt + \sigma S_t\,dW_t,$$

where r is the risk-free rate, σ the volatility, and W_t a standard Brownian motion. The value of the option at maturity T is given as:

$$S_T = S_0 \exp\Big(\big(r - \tfrac{1}{2}\sigma^2\big)\,T + \sigma\sqrt{T}\,Z\Big), \qquad Z \sim N(0,1).$$

This closed-form representation enables one-step simulation of terminal prices for European-style options, and serves as the building block for path generation in Asian and barrier options.

2.3 Option Definitions

We study three categories of options, each with distinct mathematical characteristics relevant to Monte Carlo performance.

2.3.1 European Call Option

A European call option has payoff

$$V_T^{\mathrm{EU}} = (S_T - K)^+,$$

which depends solely on the terminal price S_T. Its mathematical simplicity and closed-form valuation under the Black–Scholes model make it a standard benchmark for evaluating Monte Carlo estimators. Because the payoff is determined entirely by a single random draw of S_T, the European call serves as a fundamental baseline for assessing how variance arises in its most elementary form.

2.3.2 Asian Call Option

Asian options depend on the average prices along the entire simulated price trajectory. We use the arithmetic method since it is used more often in real life, while it has no closed form solution, giving more room to analyze it via Monte Carlo. For equally spaced times $0 < t_1 < \cdots < t_N = T$, the Asian arithmetic call payoff is

$$V_T^{\mathrm{AS}} = \left(\frac{1}{N} \sum_{i=1}^{N} S_{t_i} - K \right)^+ .$$

This payoff is path-dependent: the full set of simulated values $\{S_{t_i}\}$ contributes to the outcome rather than only the terminal value. Including this contract enables the analysis of Monte Carlo estimators in a setting where variability accumulates over many time steps, and where simulation error arises not only from terminal uncertainty but also from cumulative path fluctuations.

2.3.3 Barrier Call Option

The activation of a barrier option is determined by whether the underlying asset hits the barrier price before maturity. The down-and-in barrier call, which we use in this project, activates only if the underlying asset hits the lower barrier $H < S_0$ before maturity:

$$V_T^{\mathrm{KI}} = (S_T - K)^+ \cdot \mathbb{I}_{\left\{\min_{t \le T} S_t \le H\right\}}.$$

Here, $\mathbb{I}_{\{\cdot\}}$ denotes the indicator function of whether the asset crosses the barrier H at any point of t before maturity T. This option is both path-dependent and event-triggered: the payoff is zero unless the asset path crosses the barrier at some intermediate time. Such contracts may involve very small activation probabilities when the barrier is far from the current price. Therefore, the down-and-in call is included to represent derivatives whose valuation is strongly influenced by rare events and by the distribution of the minimum of the asset path.

3 Antithetic Variates

The Antithetic Variates method is a variance reduction technique that relies on introducing negative correlation between variables to reduce the variability of the Monte Carlo estimator. Consider two random variables X_1 and X_2 that are identically distributed, following the same distribution as a target variable X. In a standard Monte Carlo simulation, these variables would be generated independently. However, the core principle of the Antithetic Variates method is to generate X_1 and X_2 in such a way that they are dependent and negatively correlated, while preserving their individual marginal distributions.

Although this project focuses primarily on the case where inputs follow a Standard Normal Distribution—where we exploit the symmetry property by setting $X_1 = Z$ and $X_2 = -Z$—it is important to note that the theoretical framework of antithetic variates is general. It applies to any distribution where a transformation can induce negative correlation (e.g., using the Inverse Transform Sampling method with uniform inputs U and $1 - U$).

3.1 The Antithetic Estimator

To implement this method, we define a new estimator based on the average of the function evaluations at the paired points. Let $f(x)$ be the function for which we wish to compute the expectation $\alpha = \mathbb{E}[f(X)]$. The antithetic estimator for a single pair is defined as:

$$Y = \frac{f(X_1) + f(X_2)}{2}$$

3.1.1 Unbiasedness of the Estimator

A crucial property of any Monte Carlo estimator is unbiasedness. We can verify that the antithetic estimator Y is unbiased by exploiting the linearity of the expectation operator. Since X_1 and X_2 are identically distributed, they share the same expected value as the original variable X:

$$\mathbb{E}[X_1] = \mathbb{E}[X_2] = \mathbb{E}[X]$$

Consequently, the expectation of the estimator is:

$$\mathbb{E}[Y] = \mathbb{E}\left[\frac{f(X_1) + f(X_2)}{2}\right] = \frac{1}{2}\left(\mathbb{E}[f(X_1)] + \mathbb{E}[f(X_2)]\right) = \mathbb{E}[f(X)] = \alpha$$

This confirms that using the average of paired antithetic variates does not introduce any bias into the simulation.

3.2 Variance Reduction Analysis

The primary objective of this method is to reduce the variance of the estimator. To demonstrate this analytically, we compare the variance of the antithetic estimator Y against the variance of a standard Monte Carlo estimator.

Let $\sigma^2 = Var(f(X))$ be the variance of a single observation in a standard simulation. The variance of the antithetic estimator Y is derived as follows:

$$\text{Var}(Y) = \text{Var}\left(\frac{f(X_1) + f(X_2)}{2}\right)$$

Using the properties of variance for the sum of correlated variables:

$$\text{Var}(Y) = \frac{1}{4}\left[\text{Var}(f(X_1)) + \text{Var}(f(X_2)) + 2\text{Cov}(f(X_1), f(X_2))\right]$$

Since X_1 and X_2 are identically distributed, $\text{Var}(f(X_1)) = \text{Var}(f(X_2)) = \sigma^2$. Substituting this into the equation:

$$\text{Var}(Y) = \frac{1}{4}\left[2\sigma^2 + 2\text{Cov}(f(X_1), f(X_2))\right]$$

$$\text{Var}(Y) = \frac{1}{2}\sigma^2 + \frac{1}{2}\text{Cov}(f(X_1), f(X_2))$$

3.2.1 Proof of Efficiency

We now compare this with the variance of a standard Monte Carlo estimator based on two independent samples. If we were to generate two independent variables (i.e., with zero covariance), the variance of their average would simply be:

$$\text{Var}_{Indep} = \frac{\sigma^2}{2}$$

Compared with $\text{Var}(Y)$ with Var_{Indep}, we can determine the condition for variance reduction:

$$\frac{1}{2}\sigma^2 + \frac{1}{2}\text{Cov}(f(X_1), f(X_2)) < \frac{1}{2}\sigma^2$$

This inequality holds true if and only if:

$$\text{Cov}(f(X_1), f(X_2)) < 0$$

Thus, if the antithetic variates successfully induce a negative covariance between the function outputs, the variance of the antithetic estimator will be strictly less than 50% of the variance of a single standard sample (or strictly less than the variance of the average of two independent samples).

In conclusion, the Antithetic Variates method is guaranteed to reduce the variance provided that the function f preserves the negative correlation introduced in the inputs X_1 and X_2. In the context of option pricing, this condition is satisfied when the payoff function is monotonic.

3.3 Implementation and Numerical Results

3.3.1 European Call Option

To validate the theoretical results, we applied the Antithetic Variates method to price a European Call option. We utilized the following parameters: initial spot price $S_0 = 100$, strike price $K = 100$, risk-free rate $r = 5\%$, volatility $\sigma = 20\%$, and time to maturity $T = 1$ year.

The algorithm generates $M = N/2$ pairs of antithetic paths using the vectorization technique described previously. For a total of $N = 200,000$ simulations, the method demonstrates a significant efficiency gain compared to the standard Monte Carlo approach. As shown in Table 1, the Variance Reduction Factor (VRF) is approximately 2.00, meaning the standard method would require twice the computational effort to achieve the same precision.

The visual evidence of convergence is presented below. Figure 1 shows that the antithetic estimator converges faster to the true price, while Figure 2 demonstrates that the standard error remains consistently lower across different sample sizes.

Metric	**Standard MC**	**Antithetic MC**
Estimate Price	10.4515	10.4415
Standard Error (SE)	0.0233	0.0165
95% CI	(10.39, 10.52)	(10.40, 10.49)
VRF	**2.00×**	

Table 1: Comparison of Standard MC vs. Antithetic Variates ($N = 200,000$).

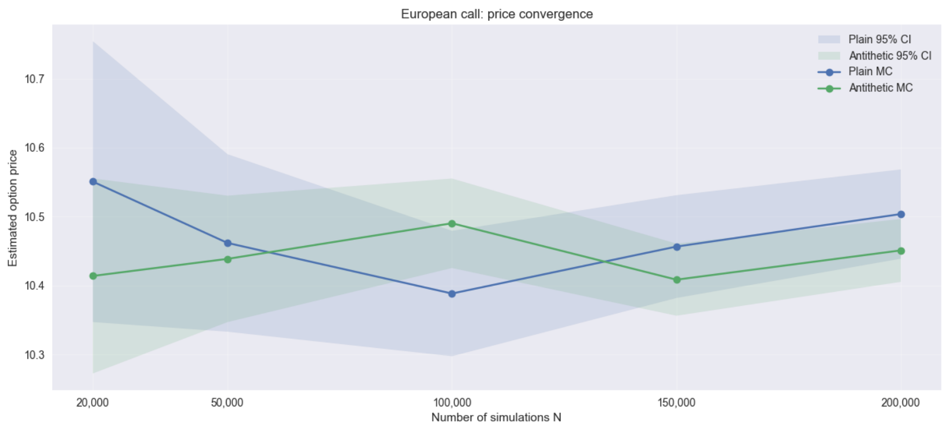

Figure 1: European Call: Price Convergence vs Number of Simulations (N).

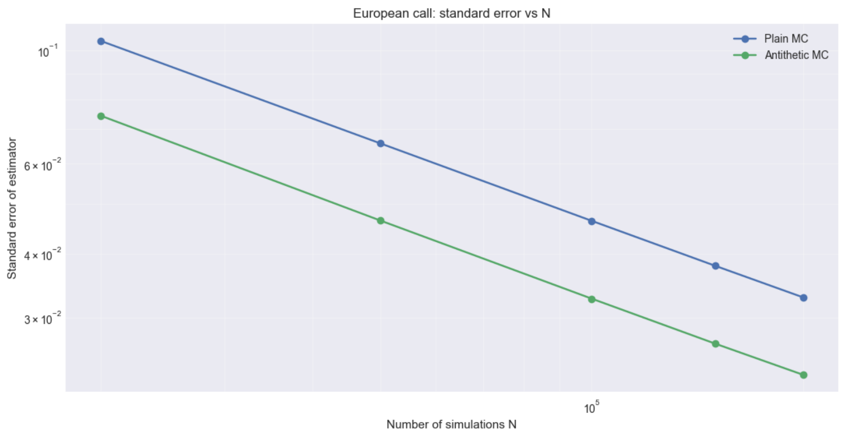

Figure 2: European Call: Standard Error Comparison.

3.3.2 Asian Call Option: Efficiency vs Moneyness

We extended the analysis to path-dependent options, specifically an Asian Call Option (Arithmetic Average), to observe how the payoff structure influences the efficiency of the antithetic variates. Fixing $S_0 = 100$, we varied the strike price K to test Deep In-The-Money (ITM), At-The-Money (ATM), and Deep Out-Of-The-Money (OTM) scenarios.

Strike (K)	Status	Payoff Shape	VRF (Efficiency)
70	Deep ITM	Linear	**50.0**×
100	At-The-Money	Convex	**2.0**×
130	Deep OTM	Flat	**1.4**×

Table 2: VRF sensitivity to option moneyness.

The results in Table 2 highlight a crucial property: when the option is Deep ITM ($K = 70$), the payoff behaves almost linearly (similar to a forward), resulting in near-perfect negative correlation and a massive VRF of 50. Conversely, for Deep OTM options ($K = 130$), the payoff is flat (zero) for many paths, reducing the covariance benefit (VRF 1.4).

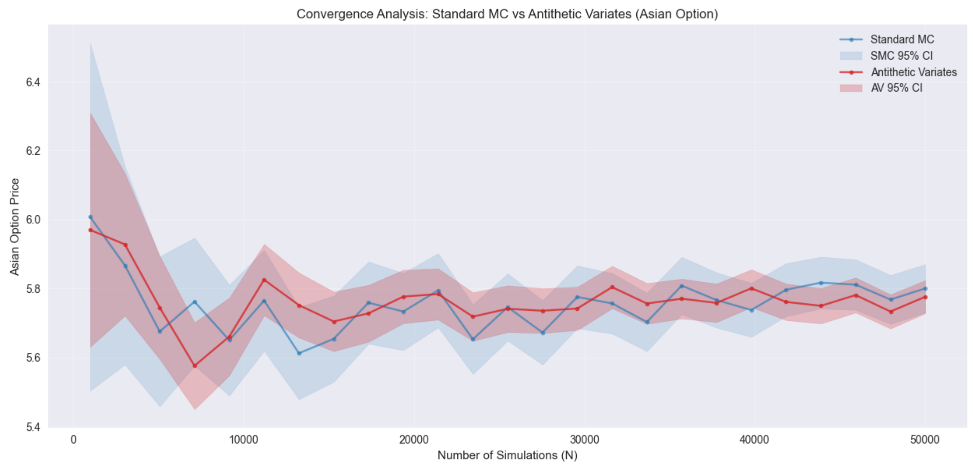

Figure 3: Asian Option Price Convergence: Standard vs Antithetic.

4 Control Variates

4.1 Concept & Theory of Control Variates

4.1.1 Motivation and Basic Mechanism

The control variate method reduces variance by incorporating information from an auxiliary random variable whose expectation is known. Suppose the target of estimation is $\mathbb{E}[X]$, and we can simulate, for each Monte Carlo path, a second variable Y that is strongly correlated with X. If Y deviates from its known mean $\mathbb{E}[Y]$ on a given simulation path, we may infer that X is likely to deviate in a similar direction. By subtracting a scaled version of this deviation, the estimator compensates for sampling noise in a systematic way.

The key idea is therefore to exploit correlation: efficient variance reduction occurs when X and Y exhibit a strong linear relationship. Because the correction term involves only $Y - \mathbb{E}[Y]$, whose expectation is zero, the adjusted estimator remains unbiased.

4.1.2 Definition of the Control Variate Estimator

Let $\overline{X}$ and $\overline{Y}$ denote the sample means of X and Y from the same n simulated paths. The control variate estimator is defined as

$$\overline{X}_{\text{CV}} = \overline{X} - b\left(\overline{Y} - \mathbb{E}[Y]\right),$$

where b is a constant determining the magnitude of adjustment. The term $\overline{Y} - \mathbb{E}[Y]$ represents the sampling error of the control variate; subtracting b times this error pulls $\overline{X}_{\mathrm{CV}}$ closer to its true mean. Since $\mathbb{E}[\overline{Y} - \mathbb{E}[Y]] = 0$, the estimator satisfies

$$\mathbb{E}[\overline{X}_{\mathrm{CV}}] = \mathbb{E}[\overline{X}] = \mathbb{E}[X],$$

so unbiasedness is preserved regardless of the choice of b. In practice, the coefficient b is later chosen to minimize variance, but the construction itself does not depend on this choice.

4.1.3 Requirements for Effective Control Variates

For the method to achieve meaningful variance reduction, several conditions must hold:

(i) **Simultaneous simulation of (X, Y):** The auxiliary variable must be generated jointly with the target variable for every Monte Carlo path, ensuring that correlation is preserved in the simulated sample.

(ii) **Known expectation of the control variate:** The value $\mathbb{E}[Y]$ must be available in closed form; otherwise, the correction term cannot be computed.

(iii) **Sufficient correlation:** A high magnitude of correlation $|\rho_{XY}|$ is essential. When the linear relationship between X and Y is weak, the variance reduction factor becomes negligible, whereas strong correlation can yield orders-of-magnitude improvement.

These conditions highlight that the effectiveness of CV relies primarily on the careful choice of a control variable that mirrors the behavior of the target payoff while remaining analytically tractable.

4.2 Variance Reduction Analysis

4.2.1 Variance of the Control Variate Estimator

To understand the efficiency of the control variate method, we analyze the variance of the adjusted estimator

$$X_{\mathrm{CV}} \;=\; X - b\big(Y - \mathbb{E}[Y]\big).$$

Since subtracting a constant does not affect variance, we have

$$\begin{aligned}\mathrm{Var}[X_{CV}] &= \mathrm{Var}\big[X - b\big(Y - \mathbb{E}[Y]\big)\big] \\ &= \sigma_X^2 - 2b\,\sigma_X\sigma_Y\,\rho_{XY} + b^2\sigma_Y^2.\end{aligned}$$

This quadratic expression in b shows how the variance depends on the strength and direction of correlation between X and Y. For values of b close to the optimal coefficient, the cross-term involving ρ_{XY} drives the reduction in variance. When the correlation is strong, even moderate deviations from the optimum still lead to substantial improvement over naive Monte Carlo.

4.2.2 Optimal Choice of the Coefficient b

The variance expression is minimized by differentiating with respect to b and setting the derivative to zero. This yields the optimal control coefficient

$$b^* \;=\; \frac{\mathrm{Cov}(X, Y)}{\mathrm{Var}(Y)} \;=\; \frac{\sigma_X\sigma_Y\rho_{XY}}{\sigma_Y^2} \;=\; \frac{\sigma_X}{\sigma_Y}\rho_{XY}.$$

This choice leverages the linear relationship between X and Y, removing from X the component that is most predictable from the behavior of Y. In numerical applications, b^* is typically replaced by its empirical estimate

$$\hat{b} = \frac{\widehat{\mathrm{Cov}}(X,Y)}{\widehat{\mathrm{Var}}(Y)},$$

which converges to the true optimal value as the number of samples increases.

4.2.3 Resulting Variance Reduction and Interpretation

Substituting b^* into the variance expression gives a closed-form reduction factor:

$$\frac{\mathrm{Var}\big[\,\overline{X}_{\mathrm{CV}}\big]}{\mathrm{Var}\big[\,\overline{X}\,\big]} = 1-\rho_{XY}^2 \qquad \Longrightarrow \qquad \mathrm{Var}\big[\,\overline{X}_{\mathrm{CV}}\big] = (1-\rho_{XY}^2)\,\frac{\sigma_X^2}{n}.$$

Thus, the achievable variance reduction depends solely on the correlation strength. When $|\rho_{XY}|$ is close to 1, the term $1-\rho_{XY}^2$ becomes small, and the estimator variance approaches zero, yielding dramatic improvements in simulation efficiency. The theoretical computational gain, often summarized as the efficiency factor, is

$$\frac{1}{1-\rho_{XY}^2},$$

indicating that even moderate correlation can produce significant speedups. Conversely, when the control variate is weakly correlated with the target variable, the reduction factor approaches 1, and CV provides little benefit. This highlights the importance of selecting a control variate that captures key structural features of the target quantity.

4.3 Implementation and Numerical Results

4.3.1 European Call Pricing

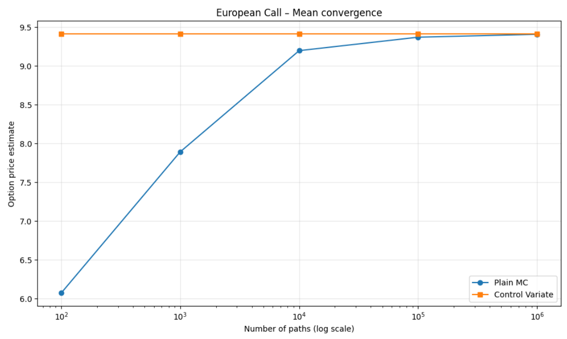

Figure 4: Mean convergence of the European call estimator under standard MC and CV.

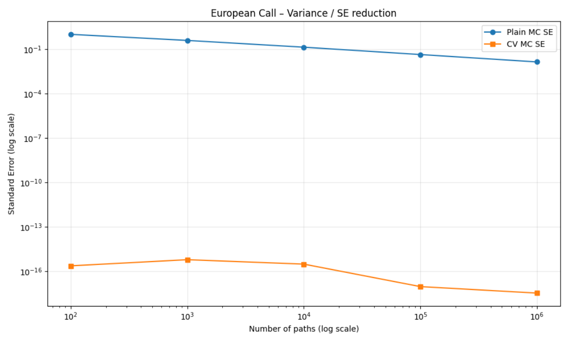

Figure 5: Variance reduction achieved by the CV estimator.

To assess the practical effect of control variates, we begin with the European call option, where the terminal stock price S_T serves as a natural control due to its strong linear relationship with the payoff $(S_T-K)^+$. Empirical results indicate that the control variate estimator achieves rapid mean convergence: while the plain Monte Carlo estimate stabilizes only after a large number of samples, the CV estimator reaches an accurate value already at sample sizes on the order of 10^2. This reflects the effectiveness of subtracting the error of S_T from the sampled payoff.

Variance reduction results mirror this behavior. Because S_T is highly correlated with the European call payoff, the sample variance of the CV-adjusted payoffs becomes extremely small, making the standard error nearly negligible relative to the plain Monte Carlo benchmark. In practice, this means that substantially fewer simulation paths are required to achieve a desired accuracy level.

4.3.2 Asian Call Pricing with EU Call as Control

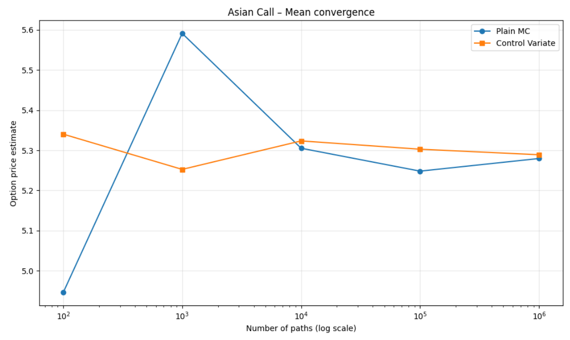

Figure 6: Mean convergence of the Asian call estimator under standard MC and CV.

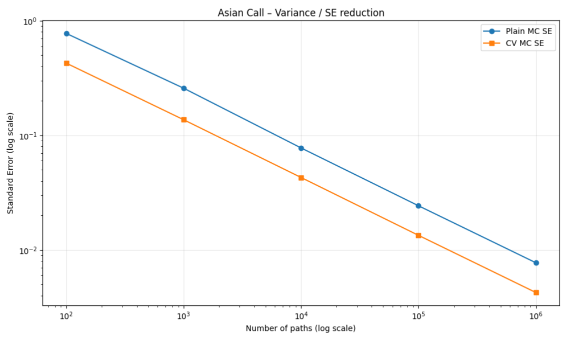

Figure 7: Variance reduction achieved by the CV estimator.

The control variate method extends naturally to the pricing of Asian arithmetic call options. Since the arithmetic average of the asset price is positively correlated with the terminal price S_T, the European call payoff can again be employed as an effective control. In simulation, the plain Monte Carlo estimator exhibits noticeable fluctuation, including significant overshooting around intermediate sample sizes (e.g., around 10^3 paths), illustrating the inherent variability arising from path-dependent payoffs.

By contrast, the control variate estimator smooths out these fluctuations. Mean convergence becomes more stable, and the variance decreases at a uniformly faster rate than under plain Monte Carlo. Although the Asian payoff differs structurally from the European payoff, the correlation is sufficiently strong to yield a meaningful reduction in estimator variance.

4.3.3 Knock-in Options Pricing with EU Call as Control

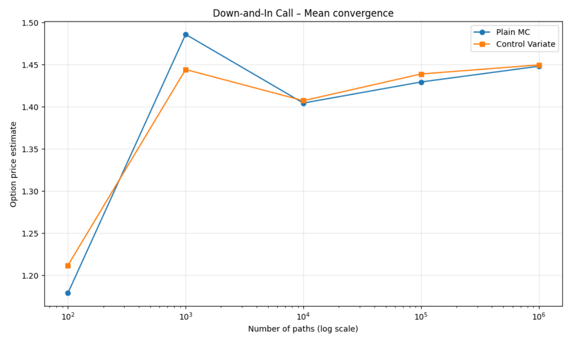

Figure 8: Mean convergence of the Knock In call estimator under standard MC and CV.

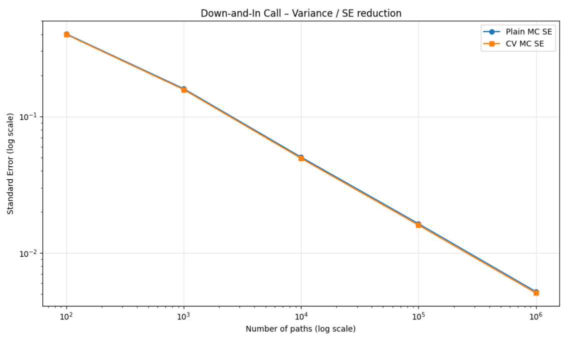

Figure 9: Variance reduction achieved by the CV estimator.

However, when applied to knock-in barrier options, the control variate approach shows clear limitations. Using the European call payoff as the control variate provides little benefit, and simulation results demonstrate that both the mean and variance of the CV estimator are nearly identical to those from plain Monte Carlo. This ineffectiveness is due to the fundamentally different structure of the rewards. Barrier activation depends on the entire path and is highly discontinuous, whereas the European call payoff depends only on the terminal stock price and varies smoothly with it. As a result, the correlation between the knock-in payoff and S_T is insufficient to drive a meaningful reduction in variance.

In such cases, more specialized control variates are required—examples include Brownian bridge–based controls or decompositions tailored to knock-in/knock-out structures—but these go beyond the scope of the present project. The numerical results highlight that the success of the CV method hinges critically on choosing a control that captures the essential variability of the target payoff.

5 Importance Sampling

5.1 Theory of Importance Sampling

5.1.1 Law of Large Numbers and Motivation for Change of Measure

The starting point for importance sampling is the classical Monte Carlo approximation of an expectation based on the Strong Law of Large Numbers. For samples $X_i \overset{\text{iid}}{\sim} p(x)$, we have

$$\mathbb{E}[f(x)] = \int_{-\infty}^{\infty} f(x)p(x)dx \approx \frac{1}{m}\sum_{i=1}^{m} f(X_i)$$

This relation shows that we can approximate the integral of $f(x)$ with respect to $p(x)$ by averaging function evaluations at i.i.d. draws from p. However, when most sampled points contribute very little to the value of the integral (for example, when f is concentrated in the tails of p), the naive Monte Carlo estimator converges slowly. The idea of importance sampling is to introduce a change of measure that assigns lower probability to such uninformative samples and higher probability to more informative ones, and then to reweight the outcomes in order to preserve unbiasedness of the estimator.

5.1.2 Formal Definition of the Importance Sampling Estimator

To formalize this idea, let p be a probability density on $\mathbb{R}^d$ and let $f : \mathbb{R}^d \to \mathbb{R}$ be a measurable function such that the following expectation is well defined:

$$I \;=\; \mathbb{E}_p[f(X)] \;=\; \int_{\mathbb{R}^d} f(x)\,p(x)\,dx < \infty.$$

We introduce another probability density q on $\mathbb{R}^d$ such that $q(x) = 0 \implies p(x) = 0$, i.e. p is absolutely continuous with respect to q. Under this condition, we can rewrite the same integral under the new density:

$$\mathbb{E}_q\left[f(X)\frac{p(X)}{q(X)}\right] = \int_{\mathbb{R}^d} f(x)\,\frac{p(x)}{q(x)}\,q(x)\,dx = \int_{\mathbb{R}^d} f(x)\,p(x)\,dx = I.$$

This suggests the following estimator when $X_1, \ldots, X_m \overset{\text{iid}}{\sim} q$.

Definition. The importance sampling estimator of I is given by

$$\hat{I}_m \;=\; \frac{1}{m}\sum_{i=1}^{m} f(X_i)\,\frac{p(X_i)}{q(X_i)}.$$

By construction, $\hat{I}_m$ is an unbiased estimator of I, and as $m \to \infty$ we have $\hat{I}_m \overset{m\to\infty}{\longrightarrow} I$ in probability. The choice of q determines the variance of this estimator and therefore its efficiency.

5.1.3 Optimal Density in the Nonnegative Case

The variance of $\hat{I}_m$ depends on how well q is adapted to the integrand $f(x)p(x)$. To make the approximation of I computationally convenient, we would like to choose q so as to minimize $\mathrm{Var}_q[\hat{I}_m]$. Consider first the special case where $f(x) \geq 0$ for all $x \in \mathbb{R}^d$. If we have

$$q(x) = \frac{f(x)p(x)}{\int_{\mathbb{R}^d} f(x)p(x)} = \frac{f(x)p(x)}{I}$$

then every sampled value of the importance sampling estimator equals I. This suggests that $\hat{I}_m$ is constant and therefore its variance is 0, which is an optimal solution in terms of variance minimization. However, this choice of q is not feasible in practice, since it requires knowledge of I itself, the very quantity we are trying to estimate.

5.1.4 General Optimal Density via Constrained Minimization

In the general case, we can view the search for an optimal importance sampling density $q(x)$ as a constrained minimization problem. Writing $\mu = I$ for brevity, we have

$$\begin{aligned}
&\min_{q(x)} \mathrm{Var}_q[\hat{I}^n] = \frac{1}{n}\left(\mathbb{E}_q\left[f(X)\frac{p(X)}{q(X)}\right]^2 - \mu^2\right) = \frac{1}{n}\int_{\mathbb{R}^d} f^2(x)\frac{p^2(x)}{q^2(x)}q(x)dx - \frac{\mu^2}{n}\\
&\text{s.t} \quad \int_{\mathbb{R}^d} q(x)dx = 1
\end{aligned}$$

which expresses the variance of the estimator in terms of q and imposes the constraint that q must be a valid probability density.
To solve this optimization problem, we introduce the Lagrangian:

$$\mathcal{L}(q(x),\lambda) = \frac{1}{n}\left(\int_{\mathbb{R}^d} f(x)^2\frac{p(x)^2}{q(x)^2}q(x)dx - \mu^2\right) - \lambda\left(\int_{\mathbb{R}^d} q(x)dx - 1\right)$$

and look for its stationary points. The first-order conditions are as follows.

$$\begin{cases}\frac{\partial\mathcal{L}}{\partial q} = \frac{1}{n}\int_{\mathbb{R}^d}\left(-f(x)^2\frac{p(x)^2}{q(x)^2} + n\,\lambda\right)dx = 0\\ \frac{\partial\mathcal{L}}{\partial\lambda} = \int_{\mathbb{R}^d} q(x)dx - 1 = 0\end{cases} \quad\Longrightarrow\quad \begin{cases} q(x) = \frac{|f(x)|\,p(x)}{\sqrt{n\,\lambda}}\\ \sqrt{n\lambda} = \int_{\mathbb{R}^d}|f(x)|\,p(x)\,dx\end{cases}$$

These conditions characterize the density form that minimizes the variance q in terms of the unknown integral of $|f(x)|$ against $p(x)$.

5.1.5 Feasibility of the Optimal Density and Open Question

The previous derivation can be summarized in the following proposition.

Proposition. The choice of $q(x)$ that minimizes $\mathrm{Var}_q[\hat{I}^m]$ is

$$q(x) = \frac{|f(x)|\, p(x)}{\int_{\mathbb{R}^d} |f(x)|\, p(x)\, dx}$$

This density is proportional to $|f(x)|p(x)$ and normalizes to one by dividing by the integral of $|f(x)|p(x)$ over $\mathbb{R}^d$.

Although this density formally minimizes the variance of the importance sampling estimator, it is again not feasible to compute in practice, because evaluating the normalizing constant

$$\int_{\mathbb{R}^d} |f(x)|\, p(x)\, dx$$

is equivalent to computing the very expectation we seek to approximate (or a closely related one). This leads to the central design question for importance sampling in applications:

Problem. How can we choose $q(x)$ in the most effective way possible, given that we only have limited information about f and p and cannot evaluate the optimal density explicitly?

The following sections address this question by developing practical constructions of q that approximate the ideal density while remaining implementable in Monte Carlo simulations.

5.1.6 Extension from Single Variables to Simulated Paths

Previously, importance sampling was formulated for a single random vector $X \in \mathbb{R}^d$ with density p and an alternative sampling density q. We now extend this framework to the case of discrete-time paths of state variables, which is the relevant setting for many path-dependent financial derivatives.

Consider a discrete path $(S(t_i))_{i=0,1,\ldots,m}$ of a Markov process, where the distribution of $S(t_{i+1})$ conditional on $S(t_i)$ is described by a transition density p_i of a random increment X_i. We may perform an importance sampling change of measure at each step by replacing p_i with another density q_i, subject to the usual absolute continuity condition. If $S(t_0)$ is given, the joint likelihood ratio for the full path is then the product of the stepwise ratios:

$$\prod_{i=1}^{m} \frac{p_i(X_i)}{q_i(X_i)} = \frac{p(X)}{q(X)}$$

where $p(X)$ and $q(X)$ denote the joint densities of the sequence of increments under the original and tilted measures, respectively.

Using this pathwise likelihood ratio, we can express expectations of a path-dependent functional f under the original measure p as expectations under the new measure q:

$$\mathbb{E}_p[f(S(t_1),\ldots,S(t_m))] = \mathbb{E}_q\left[f(S(t_1),\ldots,S(t_m)) \prod_{i=1}^{m} \frac{p_i(X_i)}{q_i(X_i)} \right]$$

This identity generalizes the single-variable importance sampling formula and provides the theoretical basis for applying importance sampling to discretized stochastic processes.

5.1.7 Instability of Likelihood Ratios for Long Horizons

While pathwise importance sampling is theoretically straightforward, it can lead to severe numerical instability when the number of time steps m is large. The source of this instability is the behavior of the product of likelihood ratios $\prod_{i=1}^{m} \frac{p_i(X_i)}{q_i(X_i)}$ as m grows. Indeed, any non-trivial change of measure ($\mathbb{P}(p_i(X_i) \neq q_i(X_i)) > 0$) on the steps leads to a highly distorted distribution of the likelihood ratio, when m is large.
The distortion is characterized by:

- A few extremely large values, occurring with very small probability.
- The most probable outcome being close to 0.

If $f(X)$ does not show a similarly degenerate behavior, importance sampling may yield infinite variance.

For small values of m, the product $\prod_{i=1}^{m} \frac{p_i(X_i)}{q_i(X_i)}$ can still behave reasonably and approximate an overall optimal change of measure, but for large m its distribution becomes highly skewed.

Example: This instability can be easily observed under the following assumptions: suppose that, for each step i, the expectation $\mathbb{E}_{q_i}[\log(p_i(X_i)/q_i(X_i))] = c$ is finite and well defined. So, by Jensen inequality, if $\mathbb{P}(p_i(X_i) \neq q_i(X_i)) > 0$:

$$c = \mathbb{E}_{q_i}\left[\log\left(\frac{p_i(X_i)}{q_i(X_i)}\right)\right] < \log\left(\int_{\mathbb{R}^d} \frac{p_i(x)}{q_i(x)} q_i(x)dx\right) = 0.$$

Combining $c < 0$ with Law of Large Numbers, for $m \to \infty$,

$$\frac{1}{m}\sum_{i=1}^{m} \log\left(\frac{p_i(X_i)}{q_i(X_i)}\right) \to c \implies \log\left(\prod_{i=1}^{m} \frac{p_i(X_i)}{q_i(X_i)}\right) \to -\infty.$$

$$\implies \prod_{i=1}^{m} \frac{p_i(X_i)}{q_i(X_i)} \to 0, \quad \text{while for any } m \quad \mathbb{E}_q\left[\prod_{i=1}^{m} \frac{p_i(X_i)}{q_i(X_i)}\right] = 1.$$

Thus, although the likelihood ratio keeps unit expectation under q, its distribution degenerates so that most realizations are extremely close to zero, with very rare but extremely large values. This behavior can lead to estimators with very high or even infinite variance if f does not counterbalance the degeneracy of the likelihood ratio.

5.1.8 Exponential Tilting and Log-Moment Generating Function

To construct practical importance sampling schemes, it is useful to work with parametric families of probability measures obtained via exponential tilting. Given a base density p on $\mathbb{R}$, exponential tilting defines a new density p_θ for each parameter θ by

$$p_\theta(x) = e^{\theta x - \psi(\theta)} p(x)$$

$$\psi(\theta) = \log \int_{-\infty}^{\infty} e^{\theta x} p(x)dx$$

The function $\psi(\theta)$ is the log moment generating function (log-MGF) of $p(x)$, and the normalizing term $\exp(\psi(\theta))$ ensures that p_θ integrates to one.

If $X_i \overset{iid}{\sim} p$, the likelihood ratio associated with tilting from p to p_θ over n samples is

$$\prod_{i=1}^{n} \frac{p(X_i)}{p_\theta(X_i)} = \exp\left(-\theta \sum_{i=1}^{n} X_i + n\psi(\theta) \right).$$

Moreover, differentiation of ψ shows that

$$\psi'(\theta) = \frac{\mathbb{E}[Xe^{\theta X}]}{\mathbb{E}[e^{\theta X}]} = \frac{\mathbb{E}[Xe^{\theta X}]}{e^{\psi(\theta)}} = \mathbb{E}[Xe^{\theta X - \psi(\theta)}] = \mathbb{E}_\theta[X]$$

so that $\psi'(\theta)$ coincides with the expectation of X under the tilted density p_θ. This identity provides a direct link between the tilting parameter θ and the shifted mean of the distribution.

5.1.9 Exponential Tilting for Normal Distributions and Pathwise Shifts

The exponential tilting construction becomes particularly tractable when the base density p is normal. Let $p(z)$ be the standard normal density and let $Z_i \overset{iid}{\sim} p$. In this case, the log-MGF is $\psi(\theta) = \theta^2/2$ and the tilted density p_θ can be computed explicitly:

$$p_\theta(z) = e^{\theta z - \theta^2/2}\, p(z) = \frac{1}{\sqrt{2\pi}}\, e^{-\frac{1}{2}(z-\theta)^2},$$

so p_θ is again normal, but shifted by θ. This means that sampling under p_θ amounts to sampling a normal random variable with mean θ and variance 1.

The same idea can be extended to a multistep path of standard normals. Considering a path $Z = (Z_1, \ldots, Z_m)$ where we apply a different shift θ_i at each step i, the likelihood ratio between the original and tilted measures is given by

$$\prod_{i=1}^{m} \frac{p(Z_i)}{p_{\theta_i}(Z_i)} = \exp\left(-\sum_{i=1}^{m} \theta_i Z_i + \frac{1}{2} \sum_{i=1}^{m} \theta_i^2 \right).$$

This representation shows that exponential tilting leads to a simple exponential form for the likelihood ratio, which can be computed efficiently even for high-dimensional paths. later, specific choices of the shifts θ_i will be designed to make rare events more likely in the simulation while preserving unbiasedness of the estimator.

5.2 Importance Sampling for Option Pricing

5.2.1 Importance Sampling for Deep Out-of-the-Money European Calls

We first illustrate the construction of an importance sampling scheme for European call options in deep out-of-the-money settings. When $S_0 \ll K$, most simulated paths under the original measure fail to produce a positive payoff, making the Monte Carlo estimator highly inefficient. To increase the frequency of informative paths, we apply exponential tilting to the standard normal increment appearing in the Black–Scholes terminal distribution.

A natural way to select the tilting parameter θ is to impose that the tilted distribution shifts the expectation of $\log S_T$ to the strike level. Specifically, the condition

$$\mathbb{E}_\theta[\log S_T] = \log S_0 + \left(r - \tfrac{1}{2}\sigma^2\right) T + \sigma\sqrt{T}\,\theta = \log K$$

leads to the explicit formula

$$\theta = \frac{\log K - \log S_0 - \left(r - \frac{1}{2}\sigma^2\right) T}{\sigma\sqrt{T}}$$

which ensures that realizations contributing to positive payoffs occur more frequently under the tilted measure. Simulation of S_T is then performed by drawing a shifted normal variable $Z + \theta$ in place of Z, and using the likelihood ratio associated with this change of measure.

Implementation: We implemented the above construction in a deep out-of-the-money scenario. For the parameter values

$$T = 10, \quad S_t = 100, \quad K = 1500, \quad r = 0.05, \quad \sigma = 0.20,$$

a comparison between standard Monte Carlo and importance sampling is obtained by repeating the simulation 100 times and computing the mean squared error relative to the analytical Black–Scholes price of \$0.0110. Figure 10 shows that exponential tilting produces a dramatic reduction in estimation error for this rare-event regime.

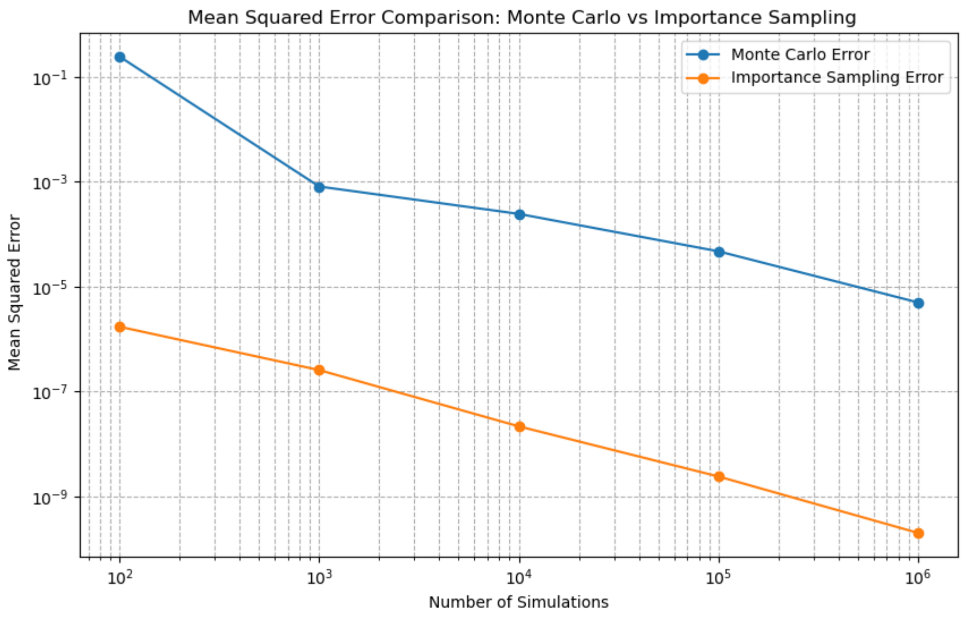

Figure 10: European Option Price Squared Error: Standard vs IS.

5.2.2 Importance Sampling for Digital Down-and-In Knock-In Options

We turn next to a more challenging application of importance sampling: digital down-and-in call options. These contracts pay P (or, in the vanilla case, $(S_T - K)^+$) only if the asset price falls below a barrier H before maturity and ends above K at time T. When $H \ll S_0$ and $K \gg S_0$, knock-in events become extremely rare under standard Monte Carlo sampling.

Assume that the underlying follows a geometric Brownian motion with discretization

$$S_n = S_0 \exp(L_n), \qquad L_n = \sum_{i=1}^{n} X_i, \qquad X_n \sim \mathcal{N}\big((r - \tfrac{\sigma^2}{2})h,\ \sigma^2 h\big)$$

for equally spaced time points $t_n = nh$. The payoff may be expressed as

$$P\,\mathbb{I}_{\{L_m > c,\ \tau < m\}}, \qquad c = \log\Big(\frac{K}{S_0}\Big), \quad -b = \log\Big(\frac{H}{S_0}\Big), \quad \tau = \inf\{n \geq 1 : L_n < -b\},$$

making explicit the two rare events: hitting the barrier, and recovering sufficiently to end above the strike.

To increase the likelihood of both the downward excursion and the upward recovery, a two-phase exponential tilting scheme is used. Before the barrier is reached, the increments $X_1, \dots, X_\tau$ are tilted with a parameter θ_- satisfying $\psi'(\theta_-) < 0$ to drive the process downward; after hitting the barrier, the increments $X_{\tau+1}, \dots, X_m$ are tilted with θ_+ such that $\psi'(\theta_+) > 0$ to guide the trajectory upward toward c.

On the event $\{\tau < m\}$, the likelihood ratio takes the form

$$\mathcal{L} = \exp\big(-\theta_- L_\tau + \psi(\theta_-)\tau\big)\ \exp\big(-\theta_+ (L_m - L_\tau) + \psi(\theta_+)(m-\tau)\big).$$

Variance reduction requires that the dependence of $\mathcal{L}$ on the random stopping time τ be eliminated as far as possible. This is achieved by imposing the condition

$$\psi(\theta_-) = \psi(\theta_+)$$

which removes the τ-dependent term in the exponent. In addition, feasibility of the path requires that the time horizon m be sufficient for the process to descend to $-b$ and ascend to c, leading to the constraint

$$\frac{-b}{\psi'(\theta_-)} + \frac{c+b}{\psi'(\theta_+)} = m.$$

For geometric Brownian motion, the log-MGF is quadratic,

$$\psi(\theta) = (r - \tfrac{\sigma^2}{2})h\theta + \tfrac{1}{2}\sigma^2 h\theta^2,$$

so that

$$|\psi'(\theta_\pm)| = \frac{2b+c}{m} \implies \theta_\pm = \left(\frac{1}{2} - \frac{\mu}{\sigma^2}\right) \pm \frac{(2b+c)}{m\sigma\sqrt{h}}$$

where μ is the drift of the log-increments.

Under exponential tilting with parameter θ, the increment distribution becomes

$$X_n \sim \mathcal{N}\big(m + v\theta,\ v\big), \qquad v = \sigma^2 h, \qquad m = (r - \tfrac{1}{2}\sigma^2)h.$$

If instead we simulate a shifted standard normal

$$Z_n^* \sim \mathcal{N}(\lambda, 1), \qquad X_n = m + \sigma\sqrt{h}\, Z_n^*,$$

then matching the means implies

$$m + \sigma\sqrt{h}\,\lambda = m + \sigma^2 h\,\theta \quad\implies\quad \boxed{\lambda = \theta\,\sigma\sqrt{h}}.$$

Thus, once θ_- and θ_+ are chosen, the corresponding shifts of the standard normals in the simulation are

$$\lambda_- = \sigma\sqrt{h}\,\theta_-, \qquad \lambda_+ = \sigma\sqrt{h}\,\theta_+.$$

Implementation: To validate the efficiency of the two-phase tilting scheme, we compared Standard Monte Carlo (Std) against the implemented Importance Sampling (IS) method on a Digital Down-and-In Call Option with a fixed payoff of 10,000. The global parameters were set to $S_0 = 95$, $r = 0.05$, and $\sigma = 0.15$.

Table 3 reports the price estimates, the variance of the IS estimator, and the Variance Reduction Factor (VRF), defined as the ratio between the variance of the Standard Monte Carlo estimator and the IS estimator.

T	m	H	K	IS Price	IS Var	VRF (Ratio)
0.25	50	94	96	3013.82	1.15e+07	**1.8**
0.25	50	90	96	426.33	4.33e+05	**9.2**
0.25	50	**85**	96	5.53	1.44e+02	**372.1**
0.25	50	90	106	13.22	7.33e+02	**162.2**
1.00	50	90	106	662.54	8.95e+05	**6.5**
1.00	50	85	96	446.13	4.76e+05	**8.5**
0.25	100	**85**	96	6.52	1.79e+02	**381.9**
0.25	100	90	106	15.50	9.03e+02	**180.1**

Table 3: Numerical comparison between Standard MC and IS for the Digital Knock-In Option. Note the extreme VRF ($> 300\times$) for the rare event cases where $H = 85$.

The results highlight that for "easy" cases (e.g., $H = 94$, close to S_0), the gain is minimal ($\approx 1.8\times$). However, for rare events ($H = 85$), where standard Monte Carlo fails to generate sufficient samples, Importance Sampling proves indispensable, reducing the variance by orders of magnitude.

5.2.3 Importance Sampling for Asian Call Options

For Asian call options, the payoff depends on the average of the underlying price along the path. Importance sampling can be adapted to this setting by applying exponential tilting to the driving Brownian increments through a linearization technique. If the pricing formula can be written as $e^{F(Z)}$ for a vector of standard normal variables $Z = (Z_1, \dots, Z_m)$, then a change of measure of the form

$$\mathbb{E}\big[e^{F(Z)}\big] = \mathbb{E}_\theta\Big[e^{F(Z)}\, e^{-\theta^\top Z + \frac{1}{2}\theta^\top\theta}\Big]$$

is considered. Rewriting the expectation in terms of $Z + \theta$ yields

$$\mathbb{E}\Big[e^{F(Z+\theta)}\, e^{-\theta^\top (Z+\theta) + \frac{1}{2}\theta^\top\theta}\Big].$$

To design θ, we approximate the exponent by linearizing $F(Z + \theta)$:

$$e^{F(Z+\theta) - \theta^\top Z - \frac{1}{2}\theta^\top\theta} \approx e^{F(\theta) + Z\,\nabla F(\theta) - \theta^\top Z - \frac{1}{2}\theta^\top\theta}$$

and choose θ to remove the dependence on Z, leading formally to the condition

$$\theta = \nabla F(\theta)^T.$$

In the case of Asian options with payoff

$$e^{F(Z)} = e^{-rT}[\overline{S}(Z) - K]^+, \qquad \overline{S}(Z) = \frac{1}{m}\sum_{j=1}^{m} S(t_j),$$

we can suppose $\overline{S} - K > 0$ and differentiate $F(Z)$:

$$\begin{cases} \frac{\partial F(Z)}{\partial z_j} = \frac{\partial \overline{S}(Z)}{\partial z_j}\, \frac{1}{\overline{S}(Z) - K} \\ \frac{\partial \overline{S}(Z)}{\partial z_j} = \frac{1}{m}\sum_{i=j}^{m} \sigma\sqrt{t_i - t_{i-1}}\, S(t_i). \end{cases}$$

Under equal time increments h, the resulting expression for the tilting vector becomes

$$\theta_j = \frac{1}{m}\left[\sum_{i=j}^{m} \sigma\sqrt{h}\, S(t_i,\theta)\right]\frac{1}{\overline{S}(\theta)-K}.$$

This leads to the recursion

$$\theta_1 = \frac{\sigma\sqrt{h}\,\overline{S}(\theta)}{\overline{S}(\theta)-K}, \qquad \theta_{j+1} = \theta_j - \frac{\sigma\sqrt{h}\,S(t_j,\theta)}{m\,(\overline{S}(\theta)-K)}, \qquad j = 1,\dots,m-1.$$

Letting $y = \overline{S}(\theta)$, we obtain the system

$$\begin{cases} g(y) = \overline{S}(\theta) - y = \frac{1}{m}\sum_{j=1}^{m} S(t_j,\theta) - y = 0 \\ \theta_1 = \frac{\sigma\sqrt{h}\,y}{y-K} \\ \theta_{j+1} = \theta_j - \frac{\sigma\sqrt{h}\,S(t_j,\theta)}{m\,(y-K)} \end{cases}$$

whose solution is found numerically by searching for the zero of $g(y)$, which can be proved to be unique.

Implementation: Applying the above tilting vector allows us to simulate the Asian option under a change of measure and compute the likelihood ratio accordingly. Numerical comparisons in the slides show the effectiveness of this approach. For instance, with parameters

$$S_0 = 100, \quad K = 100, \quad r = 0.05, \quad \sigma = 0.20, \quad T = 1, \quad m = 150$$

and $N = 50,000$ simulated paths, importance sampling achieves a variance reduction factor of approximately 13.84, reducing the standard error from 0.02205 to 0.00605. In Figure 11 we illustrate the improved confidence interval (95%) of the IS estimator relative to standard Monte Carlo simulation.

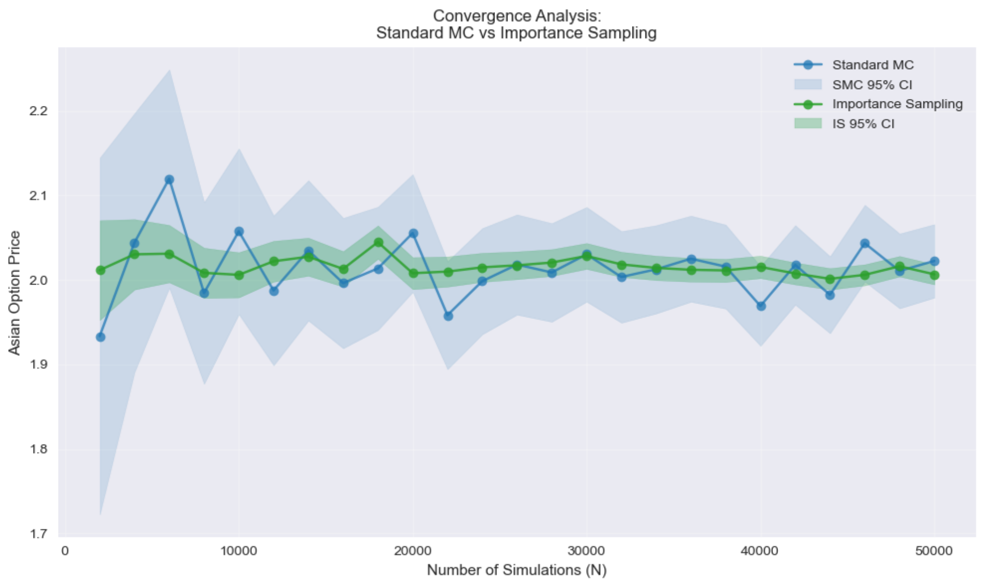

Figure 11: Asian Option Price Convergence: Standard vs IS.

5.3 Remarks

5.3.1 Conditions of Effective Importance Sampling

Importance sampling is particularly effective when the payoff is driven by rare events. Examples include deep out-of-the-money European calls, where only extreme realizations of S_T generate positive payoffs, and barrier options, where both barrier activation and ending above the strike occur infrequently. In such settings, standard Monte Carlo produces many non-informative samples, leading to slow convergence. Tilting the distribution increases the frequency of payoff-relevant scenarios while preserving unbiasedness through likelihood reweighting, thereby allocating computational effort more efficiently.

5.3.2 Limitations

The method's performance critically depends on the choice of the alternative density $q(x)$. For path-dependent problems, the likelihood ratio $\prod_{i=1}^{m} \frac{p_i(X_i)}{q_i(X_i)}$ may become unstable as m grows, collapsing toward zero for most paths and becoming extremely large for a few. This imbalance can lead to very high or even infinite variance.

Selecting tilting parameters such as θ or (θ_i) requires balancing two goals: making rare events more frequent, while preventing excessive distortion of the likelihood ratio. Because this tradeoff depends on model dynamics and payoff geometry, importance sampling often requires problem-specific design rather than a generic choice of $q(x)$.

5.3.3 Comparison with Other Variance Reduction Methods

Relative to Antithetic Variates and Control Variates, importance sampling is mostrly effective when the main difficulty is the scarcity of payoff-relevant paths. AV improves efficiency only when the payoff reacts smoothly to symmetric perturbations, and CV relies on strong correlation with an auxiliary variable—conditions that typically fail in rare-event problems.

In contrast, importance sampling modifies the sampling distribution itself, directly increasing the occurrence of critical paths. Thus, for deep OTM payoffs, discontinuous payoffs, and barrier activation events, importance sampling provides variance reductions far exceeding those of AV or CV.

6 Conclusion

This study provided a comprehensive comparative analysis of Variance Reduction Methods (VRMs) applied to European, Asian, and barrier options. By integrating theoretical derivation with empirical testing, we demonstrated that the effectiveness of any variance reduction technique is not intrinsic, but rather strictly dependent on the alignment between the estimator's design and the payoff's structural properties (convexity, path-dependence, and discontinuity).

Our investigation highlights three critical conclusions regarding the trade-off between implementation complexity and computational efficiency:

- **Antithetic Variates** serve as the most robust baseline enhancement. Although the observed VRF was consistently modest ($\approx 2\times$), the method's computational cost is negligible. It requires no prior knowledge of the payoff structure other than symmetry in the underlying distribution, making it an ideal "default" setting for standard Monte Carlo simulations.
- **Control Variates** exhibit the highest efficiency when specific structural information is available. For the Asian option, leveraging the analytical tractability of the Geometric Asian or European call yielded substantial error reduction. However, the method's failure in the Down-and-In Barrier scenario underscores a critical theoretical limitation: CV relies heavily on linear correlation. The structural mismatch between the continuous control (European option) and the discontinuous target (Barrier event) destroyed this correlation, rendering the method ineffective.
- **Importance Sampling** proved to be the only viable methodology for rare-event regimes. In scenarios where standard Monte Carlo suffers from sample impoverishment (Deep OTM and Knock-In Barrier options), IS was not merely an optimization but a necessity for convergence. The implementation of the *Two-Phase Exponential Tilting* scheme for the Barrier option was particularly notable, achieving VRFs exceeding $300\times$. This confirms that for event-driven derivatives, modifying the sampling measure is far more effective than correcting the estimator ex-post.

Table 4 synthesizes these findings, mapping each method to its optimal domain.

Method	**Best Application**	**Implementation Cost**	**Observed VRF**
Antithetic Variates	Symmetric distributions, Monotonic payoffs.	**Low** (Computational cost ≈ 0).	$\approx 2\times$
Control Variates	Path-dependent options with analytical proxies (e.g., Asian).	**Medium** (Requires high linear ρ_{XY}).	High ($\to \infty$ as $\rho \to 1$)
Importance Sampling	Rare Events (Barrier, Deep OTM) & Complex Path Structures.	**High** (Requires measure change design).	**Extreme** ($> 300\times$)

Table 4: Summary of method comparison and efficiency trade-offs.

In summary, there is no universal solution for variance reduction. For smooth, path-dependent options, Control Variates offer the optimal balance of precision and implementation effort. Conversely, for discontinuous or rare-event problems, Importance Sampling is indispensable.

7 Code

7.1 Standard Monte Carlo

7.1.1 European Options

```python
import numpy as np
from dataclasses import dataclass
@dataclass
class MCResult:
    price: float
    std_error: float
    variance: float
    ci95: tuple[float, float]
    n_paths: int

def mc_euro_bs_plain(S0: float, K: float, r: float, sigma: float, T: float,
                     n_paths: int = 100_000, option: str = "call", rng=None) -> MCResult:
    """
    Plain MC estimator for a European option under Black-Scholes.
    Returns price, std error, variance and 95% CI.
    """
    ST, _ = gbm_terminal(S0, r, sigma, T, n_paths, rng=rng)
    if option == "call":
        payoffs = call_payoff(ST, K)
    elif option == "put":
        payoffs = put_payoff(ST, K)
    else:
        raise ValueError("option must be 'call' or 'put'")

    disc_payoffs = np.exp(-r * T) * payoffs
    # Unbiased sample variance (ddof=1) to estimate Var estimator
    variance = np.var(disc_payoffs, ddof=1)
    mean = np.mean(disc_payoffs)
    se = np.sqrt(variance / n_paths)
    ci = (mean - 1.96 * se, mean + 1.96 * se)
    return MCResult(price=mean, std_error=se, variance=variance, ci95=ci, n_paths=n_paths)

    # Asian arithmetic option     Plain Monte Carlo
```

7.1.2 Asian Options

```python
def asian_arith_plain(
    S0: float, K: float, r: float, sigma: float, T: float,
    n_steps: int = 252, n_paths: int = 100_000,
    option: str = "call", include_S0: bool = True, rng=None
):
    """
    Plain Monte Carlo for arithmetic-average Asian option under GBM.
    Returns MCResult(price, std_error, variance, ci95, n_paths)     same fields as European.

    Average definition:
      if include_S0=True  : A = (1/(n_steps+1)) * sum_{j=0}^{n_steps} S(t_j)
      if include_S0=False : A = (1/n_steps)     * sum_{j=1}^{n_steps} S(t_j)
    """
    if rng is None:
        rng = np.random.default_rng()
```

```python

dt = T / n_steps
nudt = (r - 0.5 * sigma**2) * dt
sdt = sigma * np.sqrt(dt)

# Generate normal distribution for all paths and steps
Z = rng.standard_normal(size=(n_paths, n_steps))

# Simulate in log-space
log_increments = nudt + sdt * Z
log_paths = np.cumsum(log_increments, axis=1)  # shape: (n_paths, n_steps)

# Build price paths; prepend S0 if the average includes it
if include_S0:
    S = np.concatenate([np.full((n_paths, 1), S0), S0 * np.exp(log_paths)], axis
=1)
    A = S.mean(axis=1)  # average over n_steps+1 points
else:
    S = S0 * np.exp(log_paths)
    A = S.mean(axis=1)  # average over n_steps points

# Payoff
if option == "call":
    payoffs = np.maximum(A - K, 0.0)
else:
    payoffs = np.maximum(K - A, 0.0)

# Discount
disc_payoffs = np.exp(-r * T) * payoffs

variance = np.var(disc_payoffs, ddof=1)               # sample variance of
discounted payoffs
mean = np.mean(disc_payoffs)
se = np.sqrt(variance / n_paths)                      # SE of estimator
ci = (mean - 1.96 * se, mean + 1.96 * se)

return MCResult(price=mean, std_error=se, variance=variance, ci95=ci, n_paths=
n_paths)
```

7.1.3 Digital knock-in Options

```python
def standard_mc_knock_in(S0, K, H, T, r, sigma, m, N):

    Payoff: 10,000 if (min(S_t) <= H) AND (S_T > K), else 0.

    dt = T / m
    sqrt_dt = np.sqrt(dt)
    log_H = np.log(H)

    # Fixed payoff amount
    PAYOFF_AMOUNT = 10000.0

    Z = np.random.standard_normal(size=(N, m))

    # Initialize log-price paths
    # log_S has shape (N, m+1) to include t=0
    log_S = np.zeros((N, m + 1))
    log_S[:, 0] = np.log(S0)

    # Drift per time step
    drift_per_step = (r - 0.5 * sigma ** 2) * dt
    vol_per_step = sigma * sqrt_dt

    # Price evolution (Cumulative sum)
    # log S(t) = log S(0) + cumsum(drift + sigma*Z)
    increments = drift_per_step + vol_per_step * Z
    log_S[:, 1:] = np.log(S0) + np.cumsum(increments, axis=1)

```

```
        # 1. Barrier Condition (Down-and-In)
        # Check if the MINimum of the path dropped below log(H)
        min_log_S = np.min(log_S, axis=1)
        hit_barrier = (min_log_S <= log_H)

        # 2. Strike Condition (Call at maturity)
        # Check if the FINAL price is above K
        final_log_S = log_S[:, -1]
        finished_itm = (final_log_S > np.log(K))

        # 3. Joint Event
        # Both conditions must be True
        successful_paths = hit_barrier & finished_itm

        # 4. Payoff Assignment
        raw_payoffs = np.where(successful_paths,PAYOFF_AMOUNT,0.0)

        # Discounting to present value
        discount_factor = np.exp(-r * T)
        discounted_payoffs = raw_payoffs * discount_factor

        price = np.mean(discounted_payoffs)
        variance = np.var(discounted_payoffs)
        err_std = np.sqrt(variance/N)
        ci = (price - 1.96 * err_std, price + 1.96 * err_std)

        return price, variance
```

7.2 Antithetic Variate

7.2.1 European Options

```
import numpy as np
from dataclasses import dataclass
@dataclass
class MCResult:
    price: float
    std_error: float
    variance: float
    ci95: tuple[float, float]
    n_paths: int

    def mc_euro_bs_antithetic(S0: float, K: float, r: float, sigma: float, T: float,
                              n_paths: int = 100_000, option: str = "call", rng=None) ->
    MCResult:
    if n_paths % 2 != 0:
        raise ValueError("n_paths should be even for antithetic pairing.")

    if rng is None:
        rng = np.random.default_rng()

    half = n_paths // 2
    # Generate half normals and use their negatives as antithetics
    Z = rng.standard_normal(half)
    drift = (r - 0.5 * sigma**2) * T
    volT = sigma * np.sqrt(T)

    # Terminal prices for Z and -Z
    ST_pos = S0 * np.exp(drift + volT * Z)
    ST_neg = S0 * np.exp(drift - volT * Z)

    # Compute payoffs for both and average pairwise (control-style)
    if option == "call":
        g_pos = call_payoff(ST_pos, K)
        g_neg = call_payoff(ST_neg, K)
    elif option == "put":
        g_pos = put_payoff(ST_pos, K)
        g_neg = put_payoff(ST_neg, K)
```

```python
    else:
        raise ValueError("option must be 'call' or 'put'")

    # Antithetic estimator per pair
    g_hat_pair = 0.5 * (g_pos + g_neg)

    # Discount and aggregate
    disc = np.exp(-r * T)
    disc_payoffs = disc * g_hat_pair


    variance = np.var(disc_payoffs, ddof=1)         # variance of the pair averages
    mean = np.mean(disc_payoffs)
    se = np.sqrt(variance / half)                   # SE uses number of *pairs*
    ci = (mean - 1.96 * se, mean + 1.96 * se)

    return MCResult(price=float(mean), std_error=se, variance=float(variance), ci95=ci
    , n_paths=n_paths)
```

7.2.2 Asian Options

```python
import numpy as np
from dataclasses import dataclass
@dataclass
class MCResult:
    price: float
    std_error: float
    variance: float
    ci95: tuple[float, float]
    n_paths: int
def asian_arith_antithetic(
    S0: float, K: float, r: float, sigma: float, T: float,
    n_steps: int = 252, n_paths: int = 100_000,
    option: str = "call", include_S0: bool = True, rng=None
):

    if n_paths % 2 != 0:
        raise ValueError("n_paths must be even for antithetic pairing.")
    if rng is None:
        rng = np.random.default_rng()

    half = n_paths // 2
    dt = T / n_steps
    nudt = (r - 0.5 * sigma**2) * dt
    sdt = sigma * np.sqrt(dt)

    # Generate normal distribution for half the paths and build antithetic
    counterparts
    Z = rng.standard_normal(size=(half, n_steps))
    Z_bar = -Z

    # Log-increments for original and antithetic
    log_inc = nudt + sdt * Z
    log_inc_bar = nudt + sdt * Z_bar

    # Accumulate and convert to prices
    log_paths = np.cumsum(log_inc, axis=1)
    log_paths_bar = np.cumsum(log_inc_bar, axis=1)

    if include_S0:
        S = np.concatenate([np.full((half, 1), S0), S0 * np.exp(log_paths)], axis=1)
        S_bar = np.concatenate([np.full((half, 1), S0), S0 * np.exp(log_paths_bar)],
    axis=1)
        A = S.mean(axis=1)
        A_bar = S_bar.mean(axis=1)
    else:
        S = S0 * np.exp(log_paths)
        S_bar = S0 * np.exp(log_paths_bar)
        A = S.mean(axis=1)
        A_bar = S_bar.mean(axis=1)

    # Payoffs for each pair (original, antithetic)
    if option == "call":
        g = np.maximum(A - K, 0.0)
        g_bar = np.maximum(A_bar - K, 0.0)
    else:
        g = np.maximum(K - A, 0.0)
        g_bar = np.maximum(K - A_bar, 0.0)

    # Pairwise average and discount
    pair_avg = 0.5 * (g + g_bar)
    disc_pair = np.exp(-r * T) * pair_avg  # these are the i.i.d. samples for the estimator

    mean = disc_pair.mean()
    variance = disc_pair.var(ddof=1)           # variance across N/2 pair-averaged samples
    se = np.sqrt(variance / half)              # SE of the estimator uses #pairs
    ci = (mean - 1.96 * se, mean + 1.96 * se)

    return MCResult(price=mean, std_error=se, variance=variance, ci95=ci, n_paths=n_paths)
```

7.3 Control Variates

7.3.1 European Options

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from dataclasses import dataclass
from math import erf

@dataclass
class GBMParams:
    S0: float = 100.0
    K: float = 100.0
    r: float = 0.03
    sigma: float = 0.20
    T: float = 1.0
    n: int = 100_000
    seed: int = 7

# GBM paths
def gbm_paths(params: GBMParams, n_steps: int = 1) -> np.ndarray:
    rng = np.random.default_rng(params.seed)
    dt = params.T / n_steps
    Z = rng.standard_normal((params.n, n_steps))
    drift = (params.r - 0.5 * params.sigma**2) * dt
    diffusion = params.sigma * np.sqrt(dt) * Z
    log_S = np.log(params.S0) + np.cumsum(drift + diffusion, axis=1)
    return np.exp(log_S)  # shape: (n, n_steps)

#Control Variate
def mc_price_cv_paths(params: GBMParams, target_payoff_func, control_payoff_func, control_true_price: float, n_steps: int = 1):
    S_paths = gbm_paths(params, n_steps=n_steps)
    disc = np.exp(-params.r * params.T)
    payoff_target = target_payoff_func(S_paths, params)
    payoff_control = control_payoff_func(S_paths, params)
    Y = disc * payoff_target
    X = disc * payoff_control
    covXY = float(np.cov(X, Y, ddof=1)[0, 1])
    varX  = float(np.var(X, ddof=1))
    varY  = float(np.var(Y, ddof=1))
    rho_hat = covXY / (np.sqrt(varX) * np.sqrt(varY)) if varX > 0 and varY > 0 else np.nan
    b_hat   = covXY / varX if varX > 0 else 0.0
    adj = Y - b_hat * (X - control_true_price)
    mu = float(np.mean(adj))
    var = float(np.var(adj, ddof=1))
    se = float(np.sqrt(var / params.n))
    z = 1.96
    ci = (mu - z * se, mu + z * se)
    return mu, var, se, ci, b_hat, rho_hat

# Payoff functions
def eu_call_payoff_from_paths(paths: np.ndarray, params: GBMParams) -> np.ndarray:
    ST = paths[:, -1]
    return np.maximum(ST - params.K, 0.0)

# Control uses same payoff (self-control), expectation via Black Scholes
def eu_call_control_from_paths(paths: np.ndarray, params: GBMParams) -> np.ndarray:
    ST = paths[:, -1]
    return np.maximum(ST - params.K, 0.0)

# Convergence data collection
def collect_convergence_data_paths(option_name: str, params_base: GBMParams, payoff_target_func, payoff_control_func, control_true_price: float, n_paths_list, n_steps: int, B: float):
    results = {"n": [], "price_mc": [], "var_mc": [], "se_mc": [], "price_cv": [], "var_cv": [], "se_cv": []}
    for n in n_paths_list:
        p = GBMParams(S0=params_base.S0, K=params_base.K, r=params_base.r, sigma=params_base.sigma, T=params_base.T, n=int(n), seed=params_base.seed)
        price_mc, var_mc, se_mc, _ = mc_price_generic_paths(p, payoff_func=lambda paths, params: payoff_target_func(paths, params, B), n_steps=n_steps)
        price_cv, var_cv, se_cv, _, b_hat, rho_hat = mc_price_cv_paths(p, target_payoff_func=lambda paths, params: payoff_target_func(paths, params, B), control_payoff_func=lambda paths, params: payoff_control_func(paths, params), control_true_price=control_true_price, n_steps=n_steps)
        results["n"].append(int(n))
        results["price_mc"].append(price_mc)
        results["var_mc"].append(var_mc)
        results["se_mc"].append(se_mc)
        results["price_cv"].append(price_cv)
        results["var_cv"].append(var_cv)
        results["se_cv"].append(se_cv)
    return results
```

7.3.2 Asian Options

```python
@dataclass
class GBMParams:
    S0: float = 100.0
    K: float = 100.0
    r: float = 0.03
    sigma: float = 0.20
    T: float = 1.0
    n: int = 100_000
    seed: int = 7

# GBM paths function shown above

# Generic MC function shown above

# Generic Control Variate function shown above

# Payoff functions
# Target: Asian arithmetic average call
def asian_payoff_from_paths(paths: np.ndarray, params: GBMParams) -> np.ndarray:
    A = paths.mean(axis=1)
    return np.maximum(A - params.K, 0.0)
```

```python

# Control: EU call payoff from same paths
def eu_call_from_paths(paths: np.ndarray, params: GBMParams) -> np.ndarray:
    ST = paths[:, -1]
    return np.maximum(ST - params.K, 0.0)
```

7.3.3 Knock-In Option

```python
@dataclass
class GBMParams:
    S0: float = 100.0
    K: float = 100.0
    r: float = 0.03
    sigma: float = 0.20
    T: float = 1.0
    n: int = 100_000
    seed: int = 7

# GBM paths function shown above

# Generic MC function shown above

# Generic Control Variate function shown above

# Payoff functions
# Target: Down-and-In Call
def barrier_di_payoff_from_paths(paths: np.ndarray, params: GBMParams, B: float) -> np
    .ndarray:
    S_min = paths.min(axis=1)
    ST = paths[:, -1]
    hit = (S_min <= B)
    return np.where(hit, np.maximum(ST - params.K, 0.0), 0.0)

# Control: EU call payoff from same paths
def eu_call_from_paths(paths: np.ndarray, params: GBMParams) -> np.ndarray:
    ST = paths[:, -1]
    return np.maximum(ST - params.K, 0.0)
```

7.4 Importance Sampling

7.4.1 European Options

```python
import numpy as np
from scipy.stats import norm
import matplotlib.pyplot as plt
np.random.seed(42)

# Importance Sampling weight function
def Importance_Sampling_ratio(x,theta):
    return np.exp(-theta*x + 0.5*theta**2)

# Importance Sampling simulation for European call option pricing
def BS_Montecarlo_Important_Sampling(T,t,St,K,r,sigma,n: int):
    # T: Time of expiration
    # t: Current time
    # St: Underlying asset price at time t
    # K: Strike price
    # r: Risk-free interest rate (annualized)
    # sigma: Volatility of the asset's returns (annualized)
    # n: Number of Monte Carlo simulations

    theta = (np.log(K) - np.log(St) - (r - 0.5 * sigma**2) * (T-t)) / (sigma * np.sqrt
    (T-t)) # Importance sampling shift
    Z=np.random.standard_normal(n) # Generate n standard normals
    Y = Z + theta
    ST=St*np.exp((r-sigma**2/2)*(T-t)+sigma*np.sqrt(T-t)*Y) # Simulate asset prices at
     maturity
    price=np.exp(-r*(T-t))*np.sum(np.maximum(ST-K,0)*Importance_Sampling_ratio(Y,
    theta))/n
    return price
```

7.4.2 Asian Options

```python
import numpy as np
from scipy.stats import norm
import matplotlib.pyplot as plt
from scipy.optimize import brentq
np.random.seed(0)

def compute_theta(y, r, sigma, h, K, m, S0):
    """Compute the vector of shifts theta given y."""
    if y <= K:
        raise ValueError("y needs to be > K")
    theta = np.zeros(m)

    # Direct calculation of theta_1
    theta[0] = sigma * np.sqrt(h) * y / (y - K)
    S=[S0*np.exp((r-sigma**2/2)*(h)+sigma*np.sqrt(h)*theta[0])]

    # Recursive calculation of theta_{j+1}
    for j in range(m - 1):
        theta[j + 1] = theta[j] - sigma * np.sqrt(h) * S[-1] / (m * (y - K))
        S.append(S[-1]*np.exp((r-sigma**2/2)*(h)+sigma*np.sqrt(h)*theta[j+1]))
    return theta, S

def function_y(y, r, sigma, h, K, m, S0):
    """Function in y whose root we find to get optimal theta"""
    _ , S = compute_theta(y, r, sigma, h, K, m, S0)
    S_mean=np.mean(S)
    return S_mean - y

def find_optimal_theta(r, sigma, h, K, m, S0,tol):
    """Find optimal theta with Brent's method on function in y"""
    lower_bound = K + 1e-4 * max(1.0, K)
    upper_bound = max(S0 * np.exp(r*h*m),K)*10
    y_final = brentq(function_y, lower_bound, upper_bound,xtol=tol, rtol=tol, args=(r,
     sigma, h, K, m, S0))
    theta_final, _ = compute_theta(y_final, r, sigma, h, K, m, S0)
    return theta_final

def asian_IS(S0, K, r, sigma, T, n_steps, n_paths):
    # 1. Computing parameters
    dt = T / n_steps
    nudt = (r - 0.5 * sigma**2) * dt
    sdt = sigma * np.sqrt(dt)

    # 2. Computing the IS shift and likelihood ratios
    theta = find_optimal_theta(r, sigma, dt, K, n_steps, S0, tol=1e-8)
    Z = np.random.standard_normal(size=(n_paths, n_steps))
    Z += theta
    likelihood_ratio=[]
    for i in range(n_paths):
        likelihood_ratio.append(np.exp(-np.dot(theta, Z[i]) + 0.5 * np.dot(theta,
    theta)))
    likelihood_ratio = np.array(likelihood_ratio)

    # 3. Simulate paths and compute discounted payoffs
    log_increments = nudt + sdt * Z
    log_paths = np.cumsum(log_increments, axis=1)
    S = S0 * np.exp(log_paths)
    A = S.mean(axis=1)
    payoffs = np.maximum(A - K, 0.0) * likelihood_ratio
    disc_payoffs = np.exp(-r * T) * payoffs

    # 4. Compute statistics of the estimator
    variance = np.var(disc_payoffs, ddof=1)
    mean = np.mean(disc_payoffs)
    std_err = np.sqrt(variance / n_paths)
    conf_int = (mean - 1.96 * se, mean + 1.96 * se)

    return mean, std_err, variance, conf_int, n_paths
```

7.4.3 Digital Knock-In Options

```python

import numpy as np
from dataclasses import dataclass
np.random.seed(42)
import numpy as np
from numba import jit
@jit(nopython=True)

def knock_in_option_importance_sampling(S_0, K, H, T, r, sigma, m, n_sims):

    dt = T / m
    sqrt_dt = np.sqrt(dt)

    # Distances (log-space)
    b = np.log(S_0 / H)  # Distance to Barrier (must be > 0)
    c = np.log(K / S_0)  # Distance from Start to Strike

    # theta calculation
    optimal_shift = (2*b + c) / (m * sigma**2 * dt)
    theta_down_x = (1/2 - (r/sigma**2)) - optimal_shift
    theta_up_x   = (1/2 - (r/sigma**2)) + optimal_shift
    # Drifts for the Brownian Motion Z
    # Standard MC uses Z ~ N(0,1). We will use Z ~ N(shift, 1)
    shift_down = theta_down_x * sqrt_dt * sigma
    shift_up = theta_up_x * sqrt_dt * sigma

    payoffs = np.zeros(n_sims)

    for i in range(n_sims):

        # Initialize Path
        current_log_S = np.log(S_0)
        barrier_log = np.log(H)

        # Likelihood Ratio Accumulator (Sum of exponents)
        # LR = exp( - sum( theta * Z_biased - 0.5 * theta^2 ) )
        log_lr = 0.0

        barrier_hit = False

        # Time Loop
        for t in range(m):

            if not barrier_hit:
                # Phase 1: Try to hit the barrier (Drift DOWN)
                # We sample from a shifted distribution
                Z_random = np.random.normal()
                Z_biased = Z_random + shift_down  # This is the Z we use for price

                # The correction term for shift 'theta' is: -theta * Z_biased + 0.5 *
    theta^2
                log_lr += -shift_down * Z_biased + 0.5 * (shift_down**2)

                # Evolve Price
                # d(ln S) = (r - 0.5*sigma^2)dt + sigma * sqrt(dt) * Z
                current_log_S += (r - 0.5 * sigma**2) * dt + sigma * sqrt_dt *
    Z_biased
```

```

                # Check Barrier
                if current_log_S <= barrier_log:
                    barrier_hit = True

            else:
                # Phase 2: Barrier hit, now go to Strike (Drift UP)
                Z_random = np.random.normal()
                Z_biased = Z_random + shift_up

                log_lr += -shift_up * Z_biased + 0.5 * (shift_up**2)

                current_log_S += (r - 0.5 * sigma**2) * dt + sigma * sqrt_dt * Z_biased

        # 3. Calculate Payoff & Weight
        S_T = np.exp(current_log_S)

        if barrier_hit and S_T > K:
            raw_payoff = np.exp(-r * T) * 10000.0
        else:
            raw_payoff = 0.0

        # Final Weight
        likelihood_ratio = np.exp(log_lr)

        # Weighted Payoff (Unbiased Estimator)
        payoffs[i] = raw_payoff * likelihood_ratio

    # 4. Statistics
    price_estimate = np.mean(payoffs)
    variance_is = np.var(payoffs)
    se = float(np.sqrt(variance_is/n_sims))
    z = 1.96
    ci = (price_estimate-z*se,price_estimate+z*se)

    return price_estimate, variance_is
```

References

[1] P. Glasserman, *Monte Carlo Methods in Financial Engineering*. Springer-Verlag, 2004.

[2] X. Zhang, *Lectures Notes of Numerical Methods for Financial Engineering - IEDA 4520*, The Hong Kong Univesity of Science and Technology, Fall Semester 2025