Variance Reduction

Camellini Francesco, Kim Youseung, Parietti Alessandro

November 2025

Introduction

Introduction to Monte Carlo

Goal: Given a function f, compute $\alpha = E[f(X)]$.

Method:

Suppose we can draw points $X_1, \ldots, X_n$ independently from the distribution X.
The limit method estimates:

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

Theory:

The method relies on the Strong Law of Large Numbers and the Central Limit Theorem.
Although the true variance σ^2 is unknown, it can be estimated by the sample variance:

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

Variance Reduction

We introduce methods designed to reduce the variance and obtain narrower confidence intervals without increasing computational cost.

Analyzed Methods:

1. Antithetic Variates
2. Control Variates
3. Importance Sampling

Reminder: Black-Scholes Model

Under the risk-neutral measure, the asset price follows a Geometric Brownian Motion (GBM).

The value at time T, S_T, is given by:

$$S_T = S_0 \exp\left(\left(r - \frac{1}{2}\sigma^2\right)T + \sigma\sqrt{T}Z\right)$$

Where:

- S_0: Initial price
- r: Risk-free rate
- σ: Volatility
- Z: Standard Normal Random Variable, $Z \sim N(0, 1)$

Option Payoffs Summary

We compare the payoff structures of three distinct option types used in our variance reduction analysis.

Option Name	Option Type	Payoff Function (V_T)
European	Standard vanilla option.	$\max(S_T - K, 0)$
Asian	Path-dependent option.	$\max\left(\frac{1}{N}\sum_{i=1}^{N} S_{t_i} - K, 0\right)$
Knock-In	Barrier option.	$\max(S_T - K, 0) \cdot \mathbb{I}_{\{\min_t S_t \leq H\}}$

Note: $\mathbb{I}_{\{\cdot\}}$ denotes the indicator function, S_T is the spot price at maturity, and H is the barrier level.

Antithetic Variate

Introduction

We can exploit the symmetry of distributions to construct estimator.

Mechanism:

- For a standard normal distribution, $Z \sim N(0,1) \implies -Z \sim N(0,1)$.
- We can use both Z and $-Z$ to simulate the probability distributions.

The Estimator:

$$Y = \frac{f(Z) + f(-Z)}{2}$$

By exploiting the linearity of expectation and the symmetry of the distribution ($Z \overset{d}{=} -Z$), we prove that the estimator is unbiased:

$$E[Y] = \frac{1}{2}\Big(E[f(Z)] + \underbrace{E[f(-Z)]}_{=E[f(Z)]}\Big) = E[f(Z)]$$

Variance Analysis

Let us analyze the variance of the antithetic estimator $Y = \frac{f(Z)+f(-Z)}{2}$:

$$\begin{aligned}\mathrm{Var}(Y) &= \mathrm{Var}\left(\frac{f(Z)+f(-Z)}{2}\right) \\ &= \frac{1}{4}\Big[\mathrm{Var}(f(Z)) + \mathrm{Var}(f(-Z)) + 2\,\mathrm{Cov}(f(Z), f(-Z))\Big]\end{aligned}$$

Since Z and $-Z$ are identically distributed, $\mathrm{Var}(f(Z)) = \mathrm{Var}(f(-Z))$:

$$\mathrm{Var}(Y) = \frac{1}{2}\,\mathrm{Var}(f(Z)) + \frac{1}{2}\,\mathrm{Cov}(f(Z), f(-Z))$$

Conclusion

If $\mathrm{Cov}(f(Z), f(-Z)) < 0$, then:

$$\mathrm{Var}(Y) \leq \frac{1}{2}\,\mathrm{Var}(f(Z))$$

European Options: Algorithm - 1

Let N be even. We perform $M = N/2$ independent pairs of simulations.

For each $i = 1, \ldots, M$:

1. Generate $Z_i \sim \mathcal{N}(0, 1)$.
2. Calculate the two antithetic asset prices:

$$S_T^{(i,+)} = S_0 \exp\left((r - \tfrac{1}{2}\sigma^2)\, T + \sigma\sqrt{T}\, Z_i \right)$$
$$S_T^{(i,-)} = S_0 \exp\left((r - \tfrac{1}{2}\sigma^2)\, T - \sigma\sqrt{T}\, Z_i \right)$$

European Call Options: Algorithm - 2

3. Apply the payoff function $f(\cdot)$:

$$f_i^+ = f(S_T^{(i,+)}) \quad \text{and} \quad f_i^- = f(S_T^{(i,-)})$$

4. Average the Pair:

$$\hat{f}_i = \frac{f_i^+ + f_i^-}{2}$$

5. Compute the present value for the pair:

$$Y_i = e^{-rT} \hat{f}_i$$

Final Estimator: The Monte Carlo estimate is the average of the M pairs:

$$\hat{\alpha}_{AV} = \frac{1}{M} \sum_{i=1}^{M} Y_i$$

Implementation: Vectorized GBM

```python
def gbm_terminal(S0, r, sigma, T, n, rng):
    """
    Vectorized sampling of S_T.
    """
    # 1. Draw n standard normal variables
    Z = rng.standard_normal(n)

    # 2. Compute geometric motion
    drift = (r - 0.5 * sigma**2) * T
    diff  = sigma * np.sqrt(T) * Z

    ST = S0 * np.exp(drift + diff)
    return ST

```

Note: The function returns the vector of terminal prices S_T directly.

Implementation: Antithetic Pairing

```python
def mc_euro_bs_antithetic(..., n_paths, ...):
    half = n_paths // 2

    # 1. Generate half normals and use negatives
    Z = rng.standard_normal(half)

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

    # 3. Average Payoffs (Estimator Y)
    g_pos = call_payoff(ST_pos, K)
    g_neg = call_payoff(ST_neg, K)

    # The estimator is the average of the pair
    g_hat_pair = 0.5 * (g_pos + g_neg)

    # ... discount and compute variance ...

```

Results: Variance Reduction Factor (VRF)

Parameters: European Call,
$S_0 = 100, K = 100, r = 5\%, \sigma = 20\%, T = 1$.

We compared Standard MC vs. Antithetic Variates ($N = 200,000$).

Metric	**Standard MC**	**Antithetic MC**
Estimate Price	10.4515	10.4415
Std. Error (SE)	0.0329	**0.0233**
95% CI	(10.39, 10.52)	**(10.40, 10.49)**
Variance Reduction	**2.00× (VRF)**	

Efficiency Interpretation

To match this precision, Standard MC requires **twice the simulations** ($N \approx 400,000$).

European Call: Price Convergence

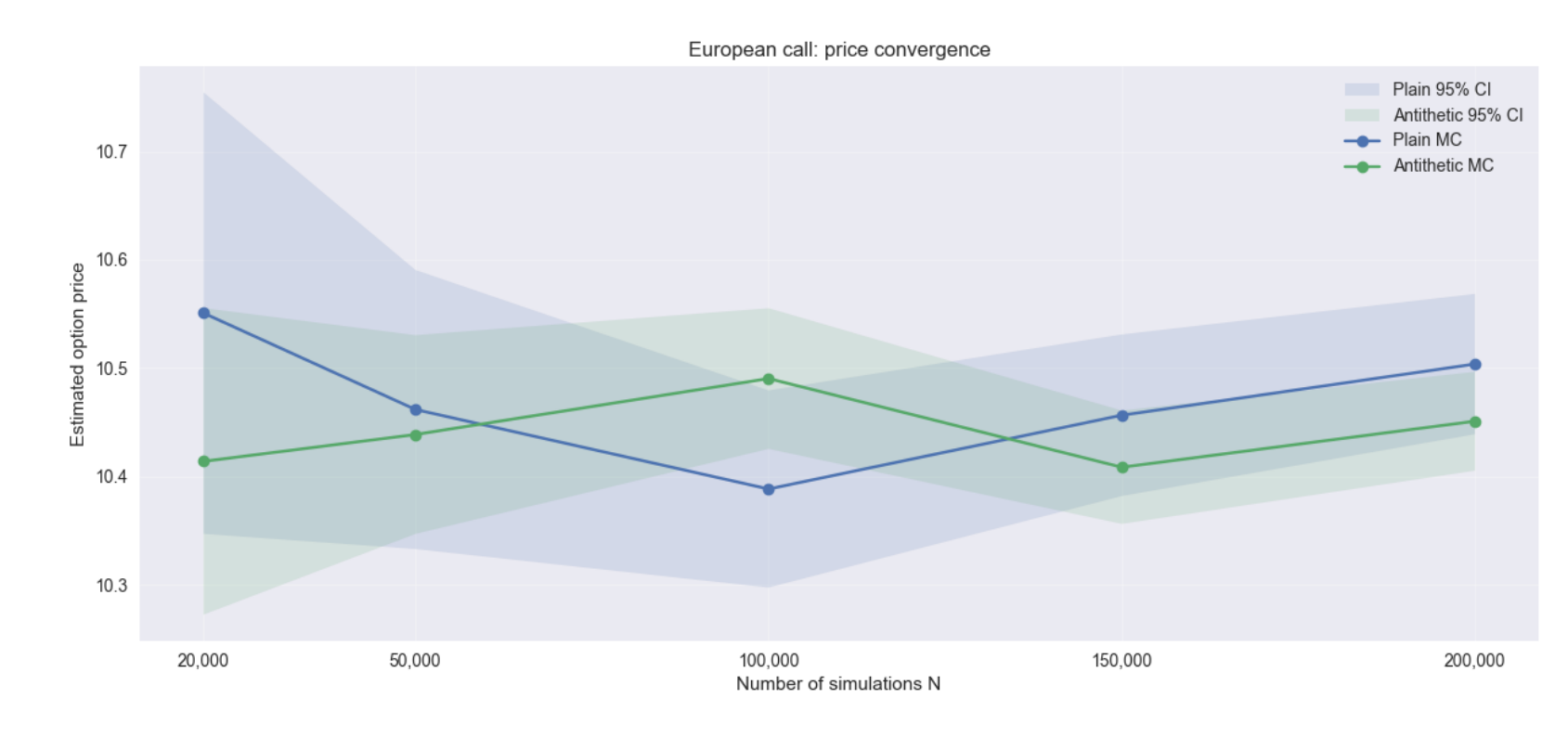


Figure 1: Plain vs Antithetic Monte Carlo: price convergence as N increases.

European Call: Standard error vs N

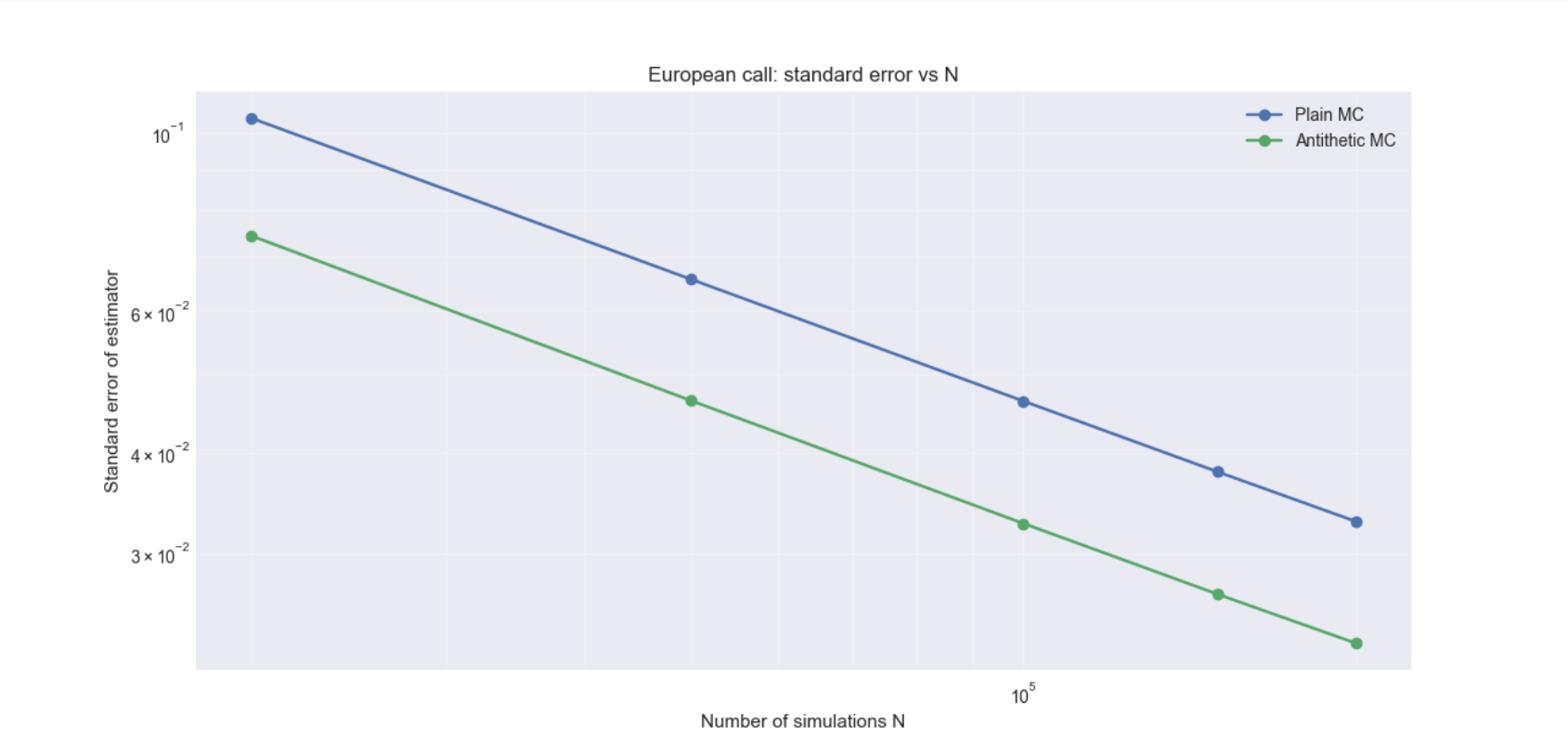


Figure 2: Standard error of the estimator: antithetic MC achieves lower variance for all N.

Efficiency vs Moneyness

We analyze the Variance Reduction Factor (VRF) for an Asian Call Option fixed at $S_0 = 100$, varying the Strike Price K.

Strike (K)	Status	Shape of Payoff	VRF (Efficiency)
$K = 70$	Deep ITM	Linear	$> 50.0\times$
$K = 100$	At-The-Money	Convex	$\mathbf{2.0}\times$
$K = 130$	Deep OTM	Flat	$1.4\times$

Observation: When the Strike is low ($K = 70$), the Call option is Deep ITM and behaves like a Forward contract. Since the payoff becomes nearly linear, the antithetic cancellation is almost perfect, resulting in massive variance reduction.

Convergence Analysis: Visual Evidence

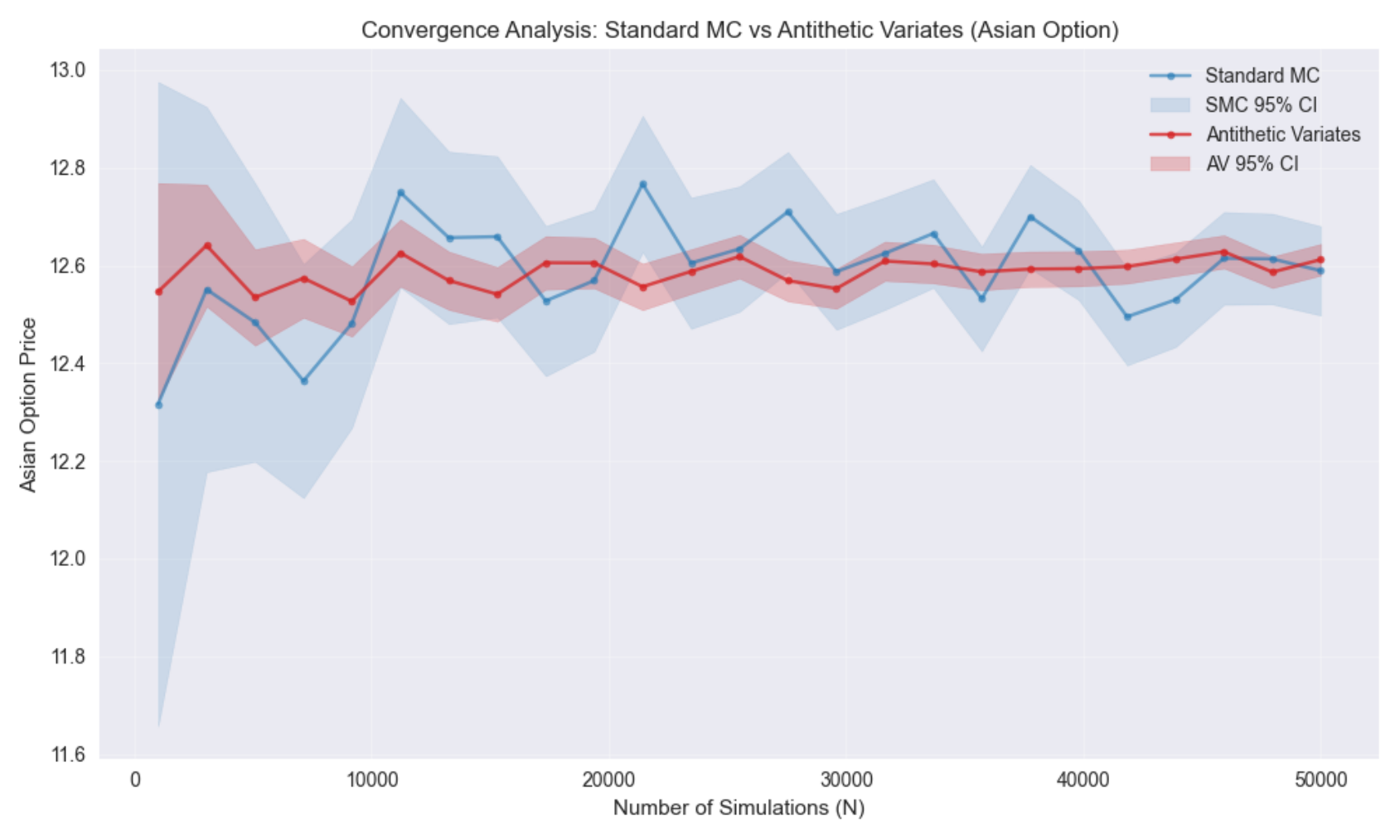


Figure 3: Data: $S_0 = 100$, $K = 90$, $N = 252$, $r = 5\%$, $\sigma = 0.2$, $T = 1$

Control Variates

Introduction

We can use an auxiliary variable Y with known $\mathbb{E}[Y]$ to reduce variance.

Mechanism:

- We know variable Y is highly correlated to the target variable X.
- By subtracting the error of Y to $\overline{X}$, we can estimate X.

The Estimator:

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

CV is unbiased as it subtracts only the error term, whose mean is zero

Requirements:

(i) Y is simulated every path, and its $\mathbb{E}[Y]$ is known.

(ii) The pairs (X_i, Y_i) are i.i.d. with finite variance.

(iii) Sufficiently high correlation $|\rho_{XY}|$ exists between X and Y.

Variance Analysis

Let us analyze the variance of CV estimator $X_{\mathrm{CV}} = X - b\big(Y - \mathbb{E}[Y]\big)$:

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

Optimal coefficient:

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

Optimal reduction: Plug in the b^*

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

Interpretation:

- Variance of X_{CV} is $(1-\rho^2)\,\sigma_X^2$.
- Variance converges to 0 as $|\rho| \to 1$.
- Computational efficiency is $\frac{1}{1-\rho^2}$

Algorithm: European Options (Control Variate)

Control: $Y_i = S_T^{(i)}$ with $\mathbb{E}[Y] = S_0 e^{rT}$

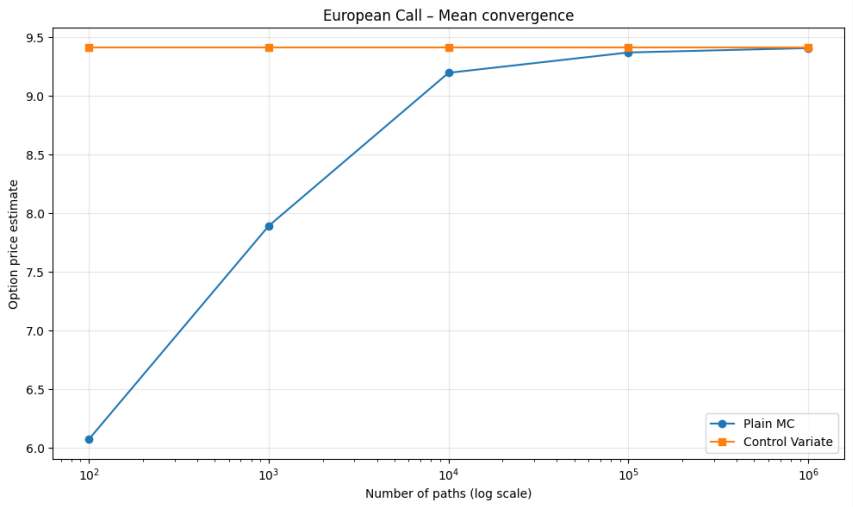
For each $i = 1, ..., N$:

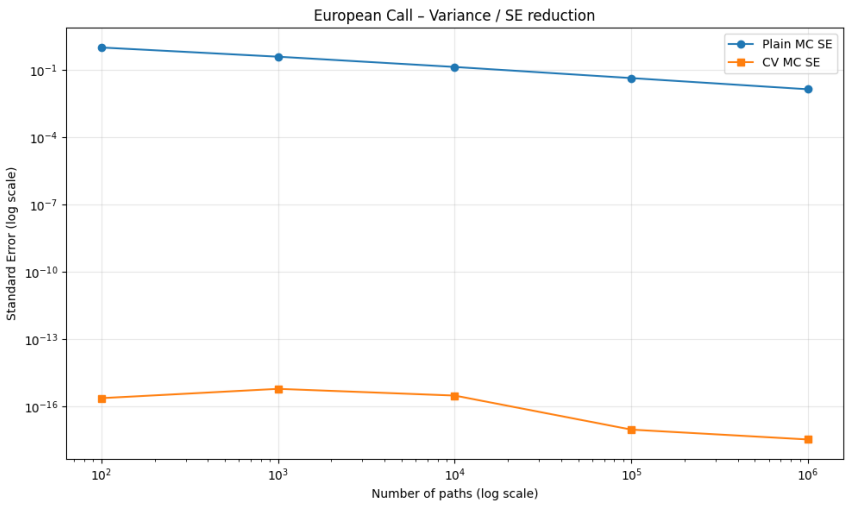
1. Generate $Z_i \sim \mathcal{N}(0,1)$, and calculate the target payoff price:
$$S_T^{(i)} = S_0 \exp\left((r - \tfrac{1}{2}\sigma^2)\, T + \sigma\sqrt{T}\, Z_i \right)$$
2. Compute payoff and control on each path:
   - Payoff: $\hat{f}_i = \max(S_T^{(i)} - K, 0)$,
   - Control variate: $Y_i = S_T^{(i)}$, $\mathbb{E}[Y] = S_0 e^{rT}$.
3. Form control-adjusted payoffs based on the control coefficient from simulated paths:
$$\hat{b} = \frac{\mathrm{Cov}(\hat{X}, Y)}{\hat{\mathrm{Var}}(Y)} \implies X_i^{CV} = X_i - \hat{b}(S_T^{(i)} - S_0 e^{rT})$$
4. Estimate the expected payoff: $\frac{1}{N}\sum_{i=1}^{N} X_i^{CV}$

Final Estimator is the mean of expected payoff discounted: $e^{-rT}\overline{X}_{CV}$

Implementation: EU Call Pricing Using Control Variate

We use the EU Call to evaluate the effect of CV by comparing the rate of mean convergence and variance reduction with the Baseline MC.

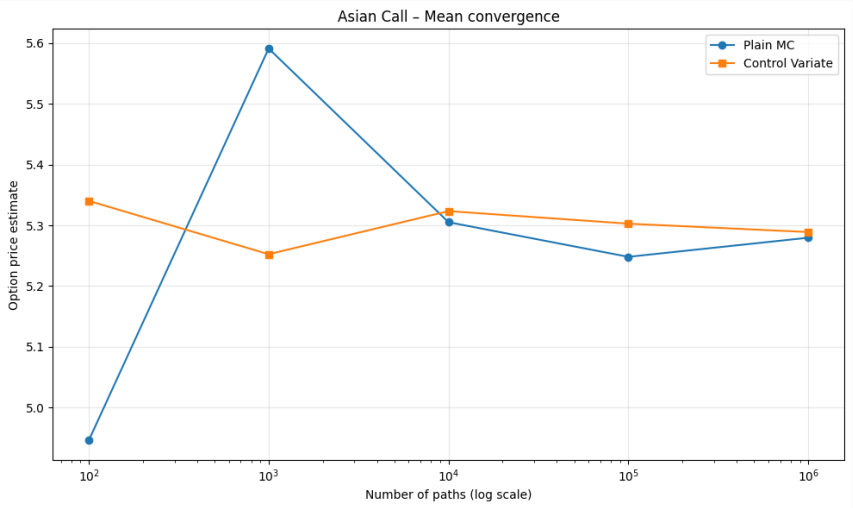


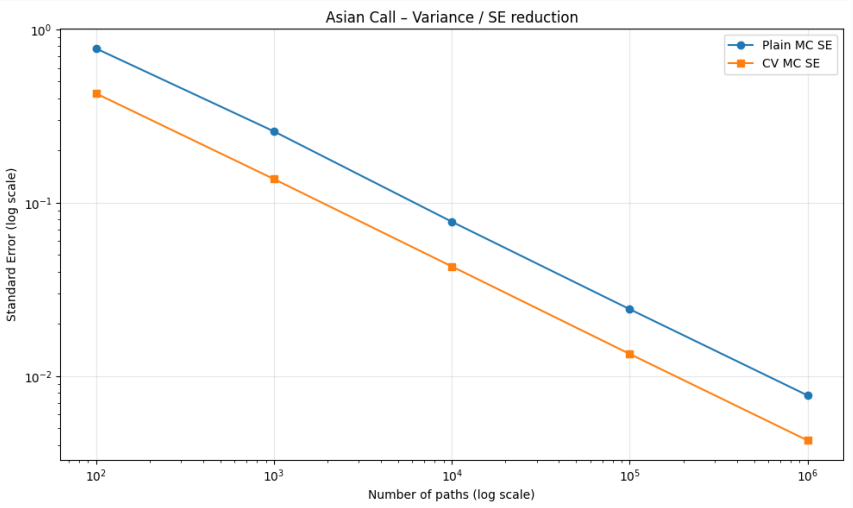


- **Mean Convergence:** Plain MC visualizes LLN while CV returns the accurate estimate from 10^2.
- **Variance Reduction:** CV Variance can be considered near 0 as SE is minimal.

Implementation: Asian Call Pricing Using CV

Asian Call Pricing is highly correlated to the EU Call Price, and thus can be used as the control.

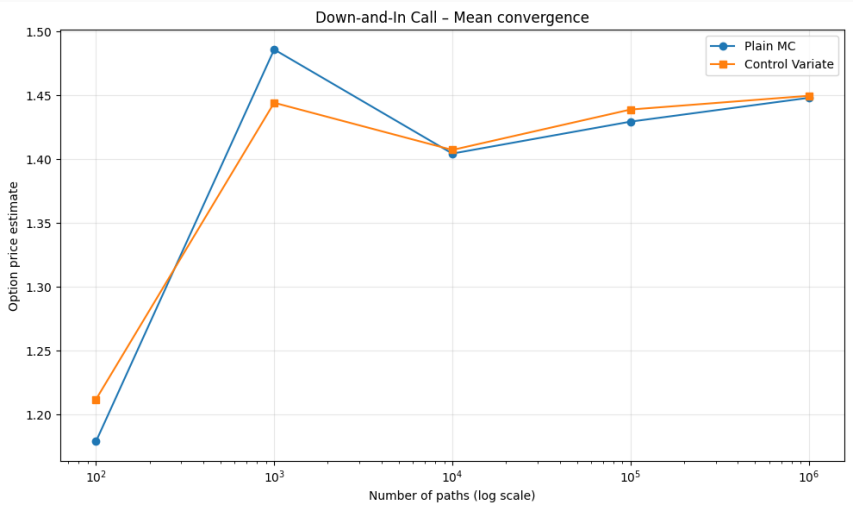


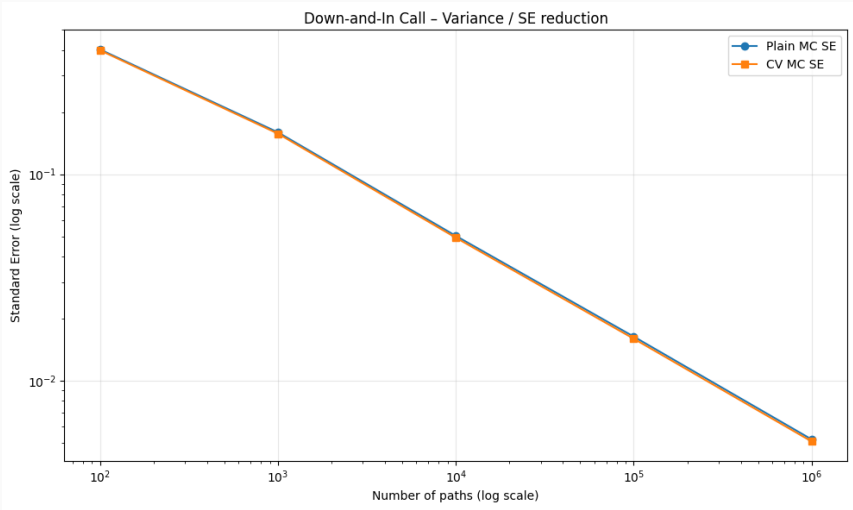


- **Mean Convergence:** Plain MC shows huge overshoot at 10^3. CV mean shows smaller fluctuation with reduced variance.
- **Variance Reduction:** Both methods return SE decreasing by $\frac{1}{\sqrt{n}}$. CV SE is strictly smaller than Plain MC SE.

Implementation: Knock-in Pricing Using CV

Once again, EU Call is used as the CV to reduce variance of Knock-in Option Price.





- **Output Analysis:** CV is ineffective for both mean & variance.
- **Issue:** Simple EU Call is not a fitting CV for Barrier Option. Additional means must be introduced to effectively control hit/no-hit (e.g., Brownian Bridge CV, Knock-in/Knock-out Decomposition).

Implementation: Code - 1

```python
# Basic EU Call Option - Implementation 1
def mc_european_call(S0, K, r, sigma, T, n, seed=42):
    rng = np.random.default_rng(seed)
    Z = rng.standard_normal(n)
    ST = S0 * np.exp((r - 0.5 * sigma**2) * T + sigma * np.sqrt(T) * Z)
    payoff = np.exp(-r * T) * np.maximum(ST - K, 0.0)
    return np.mean(payoff)

# Asian Call Option - Implementation 2
def mc_asian_arithmetic_call(S0, K, r, sigma, T, n, n_steps=252, seed=42):
    rng = np.random.default_rng(seed)
    dt = T / n_steps
    Z = rng.standard_normal((n, n_steps))
    log_paths = np.cumsum((r - 0.5 * sigma**2) * dt + sigma * np.sqrt(dt) * Z, axis=1)
    S_paths = S0 * np.exp(log_paths)
    A = S_paths.mean(axis=1)
    payoff = np.exp(-r * T) * np.maximum(A - K, 0.0)
    return np.mean(payoff)
```

Implementation: Code - 2

```python
# Knock-in Barrier Option - Implementation 3
def mc_down_and_in_call(S0, K, r, sigma, T, n, B, n_steps=252, seed=42):
    rng = np.random.default_rng(seed)
    dt = T / n_steps
    Z = rng.standard_normal((n, n_steps))
    log_paths = np.cumsum((r - 0.5 * sigma**2) * dt + sigma * np.sqrt(dt) * Z, axis=1)
    S_paths = S0 * np.exp(log_paths)
    S_min = S_paths.min(axis=1)
    ST = S_paths[:, -1]
    hit = (S_min <= B)
    payoff = np.exp(-r * T) * np.where(hit, np.maximum(ST - K, 0.0), 0.0)
    return np.mean(payoff)

# Parameters Used
S0 = 100.0      # initial price
K = 100.0       # strike price
r = 0.03        # risk-free rate
sigma = 0.20    # volatility
T = 1.0         # maturity
B = 90.0        # barrier level (for barrier options)
```

Conclusion

- CV employs another variable Y to construct $\overline{X}_{\mathrm{CV}} = \overline{X} - b(\overline{Y} - \mathbb{E}[Y])$, reducing variance with equal simulations.
- CV requirements:
  (i) Control variable Y strongly correlated with target variable X,
  (ii) Known $\mathbb{E}[Y]$.
- **Effectiveness** increases as the correlation $|\rho(X, Y)|$ grows.

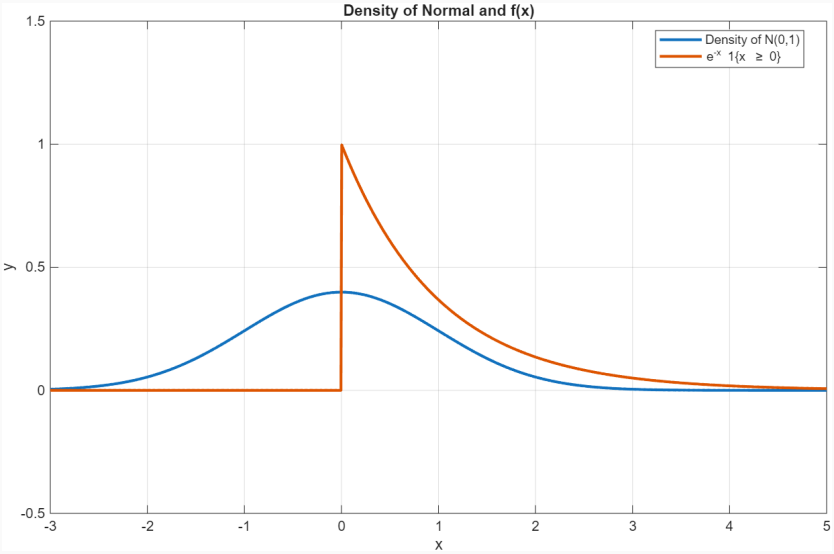
Importance sampling

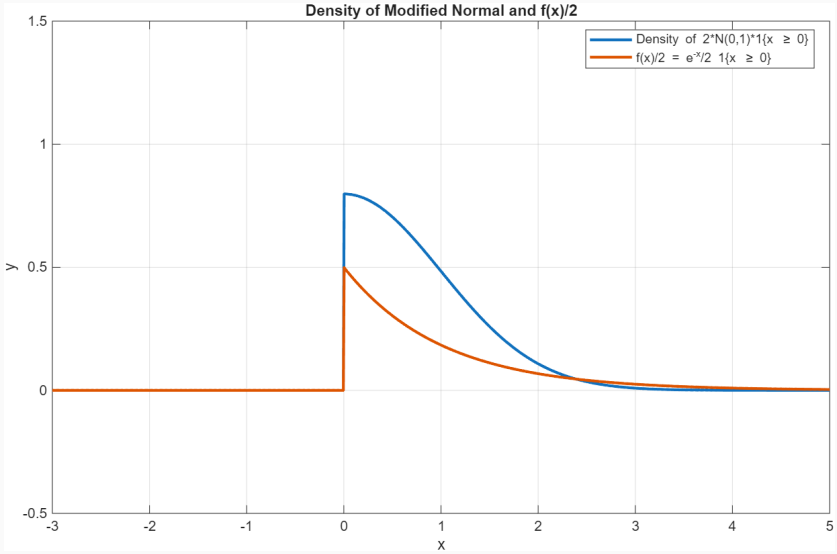
Introduction

The Strong Law of Large Numbers states that, for samples $X_i \overset{\text{iid}}{\sim} p(x)$,

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

The following example shows how a change of measure can make sampling more efficient by assigning lower probability to samples that contribute little to the estimate, and then reweighting the outcome to keep it unbiased.





Formalization

Let p be a probability density on $\mathbb{R}^d$ and let $f : \mathbb{R}^d \to \mathbb{R}$ be measurable such that is well defined

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

Let q be a probability density on $\mathbb{R}^d$ such that $q(x) = 0 \implies p(x) = 0$.

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

Definition

If $X_1, \ldots, X_m \overset{\text{iid}}{\sim} q$, the importance sampling estimator of I is

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

$\hat{I}_m$ is an unbiased estimator of I and $\hat{I}_m \overset{m\to\infty}{\longrightarrow} I$ in probability.

Variance of $\hat{I}_m$: Optimal Solution with Nonnegative $f(x)$

We want to minimize the variance of $\hat{I}_m$, in order to make the approximation of I computationally convenient.

Consider the case where $f(x) \geq 0$ for all $x \in \mathbb{R}^d$. If we have:

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

$\hat{I}_m$ is constant and therefore its variance is 0. This is clearly a optimal solution. Obviously, this approach is not feasible, as it requires the knowledge of I.

Variance of $\hat{I}_m$: Optimal Solution in the General Case

We can formulate the search of the optimal $q(x)$ as a constrained minimization problem:

$$\min_{q(x)} \operatorname{Var}_q[\hat{I}^m] = \frac{1}{n}\left(\mathbb{E}_q\left[f(X)\frac{p(X)}{q(X)}\right]^2 - \mu^2\right) = \frac{1}{n}\int_{\mathbb{R}^d} f^2(x)\frac{p^2(x)}{q^2(x)}q(x)dx - \frac{\mu^2}{n}$$

$$\text{s.t} \quad \int_{\mathbb{R}^d} q(x)dx = 1$$

We use the Lagrangian function and find its stationary points:

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

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

Variance of $\hat{I}_m$: Feasibility of Optimal Solution

Proposition

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

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

However, it is not feasible to compute this distribution, as it would be equivalent to finding I itself.

Problem

How do we choose $q(x)$ is the most effective way possible, considering our limited knowledge of the problem?

Importance Sampling with Simulated Paths

We have analyzed the case with $X \in \mathbb{R}^d$ and defined the corresponding likelihood ratio $\frac{p(X)}{q(X)}$.

We now extend this framework to analyze a discrete path $(S(t_i))_{i=0,1,\ldots,m}$ of state variables, assumed to satisfy Markov property.

- If the distribution of $S(t_{i+1})|S(t_i)$ follows the density p_i of X_i, then we can change every measure from p_i to q_i. If $S(t_0)$ is given, the resulting general likelihood ration would be:

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

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

Instability on Paths with Long Horizon - 1

Problem

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 values being close to 0.

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

For small m, the product $\prod_{i=1}^{m} \frac{p_i(X_i)}{q_i(X_i)}$ can still be a good approximation of the overall optimal change of measure.

Instability on Paths with Long Horizon - 2

Example

Consider $\mathbb{E}_{q_i}[\log(p_i(X_i)/q_i(X_i))] = c$ finite and well defined.
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 $m \to \infty$,

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

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

Exponential Tilting

We define exponential tilting as the change of measure from p to p_θ:

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

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

If $X_i \overset{iid}{\sim} p$, the likelihood ratio of the transformation is:

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

Notice that $\psi(\theta)$ is the log moment generating function of $p(x)$, so

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

Exponential Tilting on Normals

Let $p(z)$ be a standard normal and $Z_i \overset{iid}{\sim} p$. Then $\psi(\theta) = \theta^2/2$ and

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

so p_θ is a normal distribution shifted by θ.

Generalization

Considering a path $Z = (Z_1, \ldots, Z_m)$ where we apply a different shift θ_i at each step i, the likelihood ratio is given by

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

European Call Options

In European call options, we may be in deep out of money situation

- $S_0 \ll K$.
- Most Monte Carlo simulations add no contribution, if we don't change measure.

A common choice for the shift θ is obtained by imposing the condition:

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

From which we obtain:

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

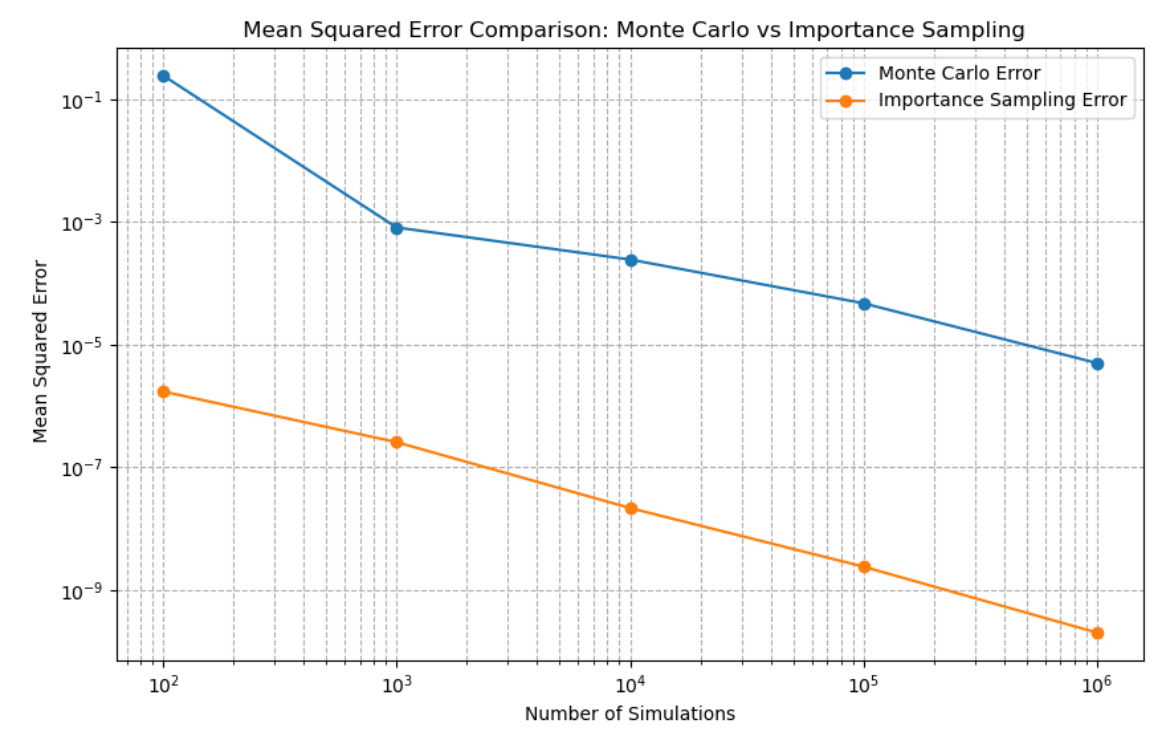
European call options: implementation

```python
# 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
```

European Call Options: Implementation

We estimate the mean squared error by comparing the analytical Black–Scholes price with estimates obtained by repeating the simulation 100 times, both with standard MC and with IS.

Variable	Value
T	10
t	0
S_t	100
K	1500
r	0.05
σ	0.20



The analytical formula yields a price of \$0.0110.

Knock-In Option: Introduction

We consider a **Down-and-In Call Option**. This is a path-dependent derivative that becomes active ("knocks in") only if a specific barrier condition is met.

The Payoff Condition: The option pays a fixed amount Q (Digital) or $(S_T - K)^+$ (Vanilla) if and only if:

1. The asset price S_t touches a lower barrier H ($S_t \leq H$) at some time $t < T$.
2. The terminal price S_T ends above the strike K ($S_T > K$).

The Rare Event Problem

If $H \ll S_0$ and $K \gg S_0$, standard Monte Carlo simulations fail. Most paths never hit H, or if they do, they fail to recover to K. The variance becomes extremely high relative to the mean.

Digital Knock-In Option - 1

The option is a digital knock-in option with payoff

$$P\,\mathbb{I}_{\{S_T>K\}}\,\mathbb{I}_{\{\min_{1\le n\le m} S_{t_n}<H\}},$$

with $0 < t_1 < \cdots < t_m = T$, where S is the underlying asset, K the strike, H the barrier and P the possible payoff.

The goal is to use importance sampling in order to make knock-in events less rare.

Suppose the underlying asset S follows a Geometric Brownian Motion $\mathrm{GBM}(r, \sigma^2)$ with equally spaced time points $t_n = nh$. Then the BM is

$$S_n = S_0\,\exp(L_n), \qquad L_n = \sum_{i=1}^{n} X_i$$

$$X_n \sim \mathcal{N}\big((r - \tfrac{\sigma^2}{2})h,\ \sigma^2 h\big)$$

Digital Knock-In Option - 2

The payoff can be written as: $P\,\mathbb{I}_{\{L_m > c,\ \tau < m\}}$, with

$$c = \log\left(\frac{K}{S_0}\right), \qquad -b = \log\left(\frac{H}{S_0}\right), \qquad \tau = \inf\{n \geq 1 : L_n < -b\}.$$

If b or c is large, the probability of $\{L_m > c,\ \tau < m\}$ is very small. To increase this probability:

- Before the barrier: we tilt the increments $X_1, \ldots, X_\tau$ with a parameter θ_- (so that $\psi'(\theta_-) < 0$) to make the walk more likely to move *down* toward $-b$ and hit the barrier.
- After the barrier: we tilt the remaining increments $X_{\tau+1}, \ldots, X_m$ with a parameter θ_+ (so that $\psi'(\theta_+) > 0$) to drive the walk *up* toward c and increase the probability of ending in-the-money.

On the event $\{\tau < m\}$, the likelihood ratio for this two–phase change of measure is

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

Digital Knock-In Option - 3

1. Eliminating Randomness (Variance Reduction): The likelihood ratio depends on the random stopping time τ:

$$\mathcal{L} = \exp\Big((\theta_+ - \theta_-)\underbrace{L_\tau}_{\approx -b} - \theta_+\underbrace{L_m}_{\approx c} + (\psi(\theta_-) - \psi(\theta_+))\tau + m\psi(\theta_+)\Big)$$

To remove this dependency (and reduce variance), we impose:

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

2. Path Constraints: The total time m must be sufficient to travel down to $-b$ and back up to c:

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

For GBM, $\psi(\theta) = (r - \frac{\sigma^2}{2})h\theta + \frac{1}{2}\sigma^2 h\theta^2$ is quadratic and so:

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

Digital Knock-In Option - 4

Under exponential tilting with parameter θ, the tilted law of X_n is

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

If instead we simulate a shifted normal

$$Z_n^* \sim \mathcal{N}(\lambda, 1), \qquad X_n = m + \sigma\sqrt{h}\, Z_n^*$$
$$\Longrightarrow\ X_n \sim \mathcal{N}\big(m + \sigma\sqrt{h}\,\lambda,\ \sigma^2 h\big).$$

Matching the means gives

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

Normal shifts for the two phases

Once θ_- and θ_+ are chosen for the increments X_n, the corresponding shifts of the standard normals are

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

Digital Knock-in Option: Illustration

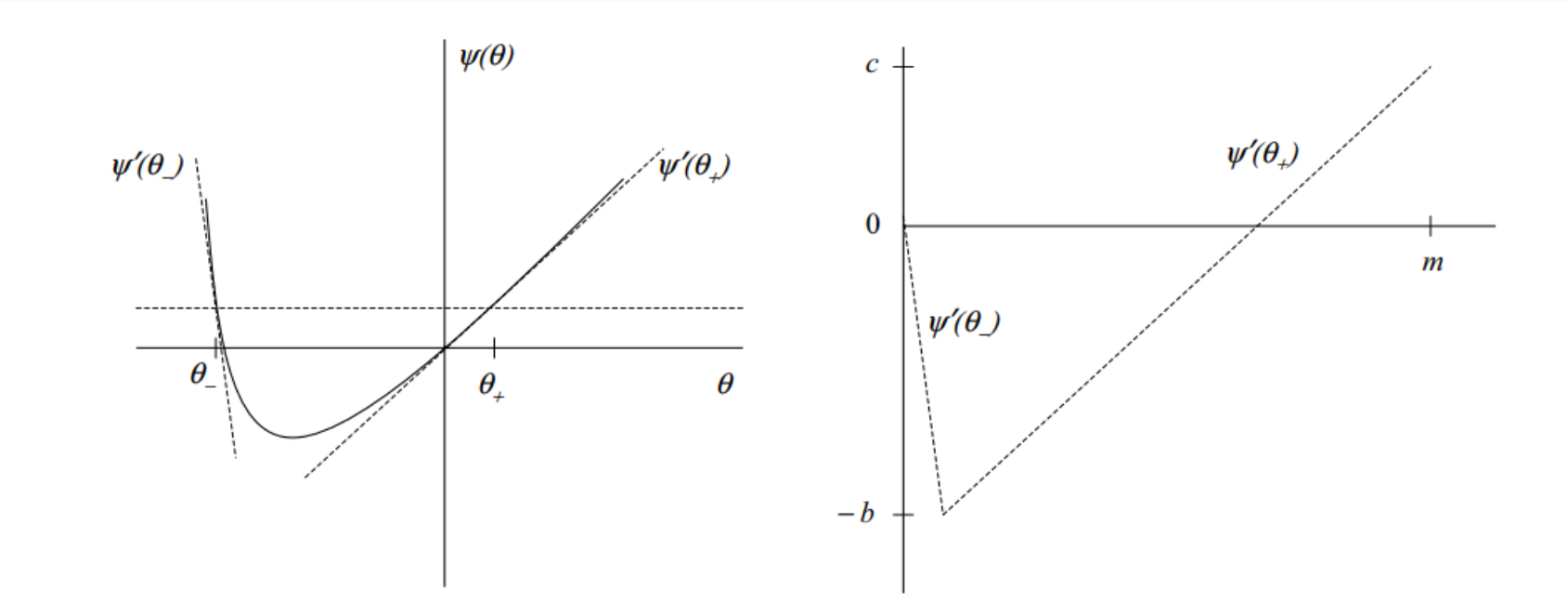


Figure 4: Figure from Glasserman, p. 275

Digital Knock-in Option: Implementation - Setup

```
def knock_in_option_is(S0, K, H, T, r, sigma, m, n_sims):
    # 1. Constants & Log-Distances
    dt, sq_dt = T / m, np.sqrt(T / m)
    b, c = np.log(S0 / H), np.log(K / S0)
  # We need to cover distance 2b + c (Down to H, back to S0, up to K
    )
    optimal_shift = (2*b + c) / (m * sigma**2 * dt) # Approximation

    # Drifts for the Brownian Motion X (not the price S)
    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 (not the price S)
    # 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)
    log_H = np.log(H)
```

Digital Knock-in Option: Illustration - Simulation Loop

```
    # 3. Simulation
    for i in range(n_sims):
        log_S, log_lr, hit = np.log(S0), 0.0, False

        for _ in range(m):
            # Dynamic Drift: Switch strategy if barrier is hit
            drift = shift_up if hit else shift_down

            # Sample biased increment
            Z = np.random.normal() + drift

            # Update Likelihood Ratio & Price
            log_lr += -drift * Z + 0.5 * drift**2
            log_S  += (r - 0.5*sigma**2)*dt + sigma*sq_dt*Z

            if log_S <= log_H: hit = True

        # 4. Final Estimator
        payoff = 10000.0 if (hit and np.exp(log_S) > K) else 0.0
        payoffs[i] = payoff * np.exp(log_lr)

    return np.mean(payoffs), np.var(payoffs)
```

Digital Knock-in Option: Numerical Results

We compared Standard Monte Carlo (Std) vs Importance Sampling (IS) on a Digital Option (Payoff = 10,000).

Parameters: $S_0 = 95, r = 0.05, \sigma = 0.15$.

T	m	H	K	IS Price	IS Var	Ratio (VRF)
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**

Conclusion: As the event becomes rarer (e.g., Barrier $H = 85$), the Variance Reduction Factor (VRF) explodes (> 370), proving the method's extreme efficiency for rare events.

Change of Drift: Linearization

Let $Z = (Z_1, \ldots, Z_n)$ be a standard multi-normal variable and $e^{F(Z)}$ a path dependent pricing formula.

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

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

We want a constant inside the expectation. We linearize:

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

Therefore we choose:

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

Asian Call Options - 1

For Asian Options we have:

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

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

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

Considering constant intervals h, we find θ:

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

Asian Call Options - 2

The previous relation leads to the recursion:

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

We can set $y = \overline{S}(\theta)$, and then find $\theta_j(y)$, obtaining the system:

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

This system can be solved numerically, by finding the zero (which can be proved to be unique) of the function $g(y)$.

Asian Call Options: Implementation

```
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

```

Asian Call Options: Implementation

```
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)
```

Asian Call Options: Implementation

```
    # 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)

```

Asian Call Options: Implementation results

Parameters: $S_0 = 100,\ K = 100,\ r = 5\%,\ \sigma = 20\%,\ T = 1.$

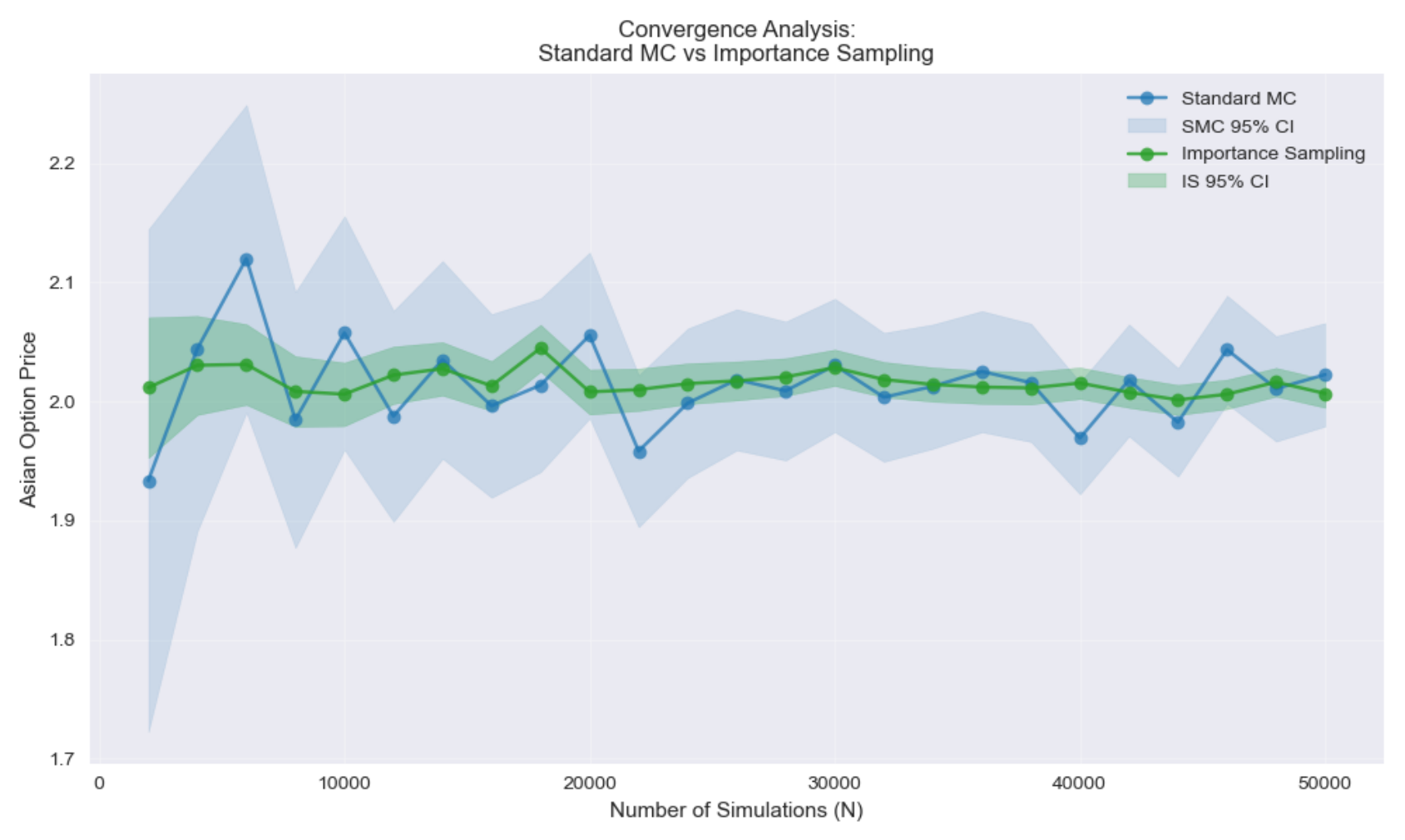
We compare Standard MC vs. Importance Sampling ($N = 50,000$).

Metric	**Standard MC**	**IS**
Estimate Price	1.9289	2.0192
Std. Error (SE)	0.02205	0.00605
95% CI	(1.8857, 1.9721)	(2.0073, 2.0311)
Variance Reduction	**13.84× (VRF)**	

Efficiency Interpretation

To match this precision, Standard MC requires $\approx 700,000$ simulations.

Asian Call Options: Implementation results



Conclusion

Summary & Method Comparison

Method	Best Application	Implementation Cost	Observed VRF
Antithetic Variates	Symmetric distributions, Monotonic payoffs (e.g., Standard Calls).	**Low** (Only requires sign flip).	$\approx 2\times$
Control Variates	When a correlated analytical formula exists (e.g., Geom. Asian, Euro Call).	**Medium** (Requires $\mathbb{E}[Y]$ and ρ_{XY}).	High ($\to \infty$ as $\rho \to 1$)
Importance Sampling	Rare Events: Deep OTM options, Barrier options (Knock-in).	**High** (Math-heavy, optimization required).	**Extreme** ($> 300\times$)

Context Matters: CV failed for the Barrier option due to low correlation, whereas IS proved essential for convergence in rare event scenarios.

Thank you!