Imperial College London

Quantum Computing

Imperial College London

Department of Mathematics

Assessed Coursework

Authors:
Charles Garrisi (CID 06009336)
Arthur Nahmias (CID 06020137)

Professors:
Dr. Jack Jacquier

January 18, 2025

Contents

Introduction

Numerical approaches are regularly used for option pricing in the industry as models that provide closed-form solutions like Black–Scholes–Merton actually rely on simplifying assumptions that are often not verified in practice. Monte Carlo (MC) simulation is particularly popular due to its flexibility and ability to handle high-dimensional or path-dependent payoffs. However, classical MC pricing is typically computationally expensive when pricing complex options. Minor improvements in simulation efficiency can actually translate into substantial savings in time and computational resources for practitioners.

In this work, we present a quantum-enhanced algorithm from (An *et al.*, 2023) [1] aimed at accelerating classical Monte Carlo methods for option pricing under stochastic differential equations (SDEs). While standard MC estimates the expectation $\mathbb{E}[Y]$ of the SDE solution Y numerically the proposed method leverages the strengths of quantum amplitude estimation to achieve a better error dependence, thus reducing the sampling overhead that usually limits classical MC approaches. This approach is promising for improving the speed and scalability of simulations used in derivative pricing, risk management, and other computationally intensive tasks in quantitative finance.

1 Overview of the problem

Let X_t be a stochastic process with general drift $\mu(t, X_t)$ and volatility $\sigma(t, X_t)$, described by the following stochastic differential equation

$$dX_t = \mu(t, X_t)\, dt + \sigma(t, X_t)\, dW_t,$$

where W_t is a standard Brownian motion. As we will see more in detail in the following section, the Monte Carlo method enables us to compute $\mathbb{E}[P(X) \mid X \in \pi_0]$, where $P(X)$ is a payoff function and π_0 is a given state space constraint. Let's consider the Black-Scholes model in option pricing, where the stock price S_t follows a geometric Brownian motion:

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

with r as the risk-free interest rate and σ as the volatility. Under those settings, the price of the option V_t satisfies the Black-Scholes equation:

$$\frac{\partial V}{\partial t} + rS\frac{\partial V}{\partial S} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} - rV = 0 \qquad (*)$$

By the Feynman-Kac theorem, we know that the solution of the Black-Scholes PDE is expressed as:

$$V(t, S) = \mathbb{E}\left[e^{-r(T-t)}\phi(S_T) \mid S_t = S\right] \qquad (**)$$

where $\phi(S_T)$ is the payoff of the option at maturity T. While Monte Carlo method is regularly used to compute this expectation, We'll see in the following how Quantum Monte Carlo enables us to enhance its computation speed.

2 Monte Carlo method

2.1 Principle

Monte Carlo simulation is a fundamental method for numerically estimating the fair value of options and other derivatives by sampling from the underlying stochastic processes. Consider the previous SDE

$$dX_t \;=\; \mu\big(X_t, t\big)\, dt \;+\; \sigma\big(X_t, t\big)\, dW_t,$$

where W_t is a standard Brownian motion, and μ, σ are drift and diffusion coefficients, respectively. Let P be a payoff function evaluated at time T. The goal is to approximate $\mathbb{E}_{\mathbb{P}}[P(X_T)]$, where $\mathbb{P}$ is the probability measure driving the stochastic dynamics of X_t – note that when an option payoff is delivered at maturity T at a constant risk-free rate r, its fair price is given by $e^{rT}\,\mathbb{E}_{\mathbb{P}}\big[P(X_T)\big]$. In practice, Monte Carlo implements a discrete-time approximation and samples multiple paths from the induced distribution to estimate this expectation.

A classical MC procedure simulates N independent realizations of the underlying process, denoted by $\{X_M^{(i)}\}_{i=1}^N$ when discretizing X_t with M time steps. By averaging the N independent samples, one can compute

$$\widehat{P}_{\mathrm{MC}} \;=\; \frac{1}{N}\sum_{i=1}^{N} P\big(X_M^{(i)}\big)$$

By the law of large numbers, $\widehat{P}_{\mathrm{MC}}$ converges to $\mathbb{E}_{\mathbb{P}}\big[P(X_M)\big]$ as $N \to \infty$. However, the true objective is $\mathbb{E}[P(X_T)]$. We'll see in the following how to reduce this discrepancy.

2.2 Numerical Schemes

The discrete approximation X_M arises from a chosen numerical scheme for the SDE. Many methods differ in their convergence properties. A scheme has *weak order* α if it approximates the distribution of X_t to $O((\Delta t)^\alpha)$, sufficient for many payoff-driven applications. A scheme has *strong order* r if it remains pathwise close in an L^p sense, which can be advantageous for path-dependent payoffs or refined variance reduction.

Definition : Strong order r

A numerical scheme has *strong order* r if its approximation $\hat{X}_k$ satisfies

$$\mathbb{E}\Big[\sup_{0\le kh\le T}\big|\hat{X}_k - X_{kh}\big|^p\Big] \;\le\; C\,h^{rp},$$

for all $p \ge 1$. Here, X_t is the exact solution of the SDE, $\hat{X}_k$ is the numerical approximation at discrete times of size h, and C is a constant independent of h.

Euler–Maruyama Scheme

Euler–Maruyama is a basic numerical method of strong order $r = 0.5$. It updates

$$X_{k+1} = X_k + \mu(X_k, t_k)\,h + \sigma(X_k, t_k)\,\Delta W_k,$$

where ΔW_k are independent $\mathcal{N}(0, h)$ increments. Although its strong order is only 0.5, it often achieves weak order 1, making it popular in finance for basic payoff computations.

Milstein Scheme

The Milstein scheme has strong order $r = 1$ under suitable smoothness assumptions. It refines Euler–Maruyama by adding a term involving the derivative of σ:

$$X_{k+1} = X_k + \mu(X_k, t_k)\,h + \sigma(X_k, t_k)\,\Delta W_k + \tfrac{1}{2}\,\sigma(X_k, t_k)\,\frac{\partial\sigma}{\partial X}(X_k, t_k)\,\big((\Delta W_k)^2 - h\big).$$

This extra term cancels out part of the local approximation error, thereby increasing the strong order to 1.

2.3 Total Complexity

The mean-squared error (MSE) of $\widehat{P}_{\mathrm{MC}}$ separates into a *bias* term due to time discretization and a *variance* term due to finite sampling:

$$\mathrm{MSE}\big(\widehat{P}_{\mathrm{MC}}\big) = \big|\mathbb{E}[P(X_T)] - \mathbb{E}[P(X_M)]\big|^2 \;+\; \mathrm{Var}\big[\widehat{P}_{\mathrm{MC}}\big].$$

The bias typically scales as $M^{-\alpha}$, where α is the *weak convergence order* of the SDE solver. More timesteps ($M \to \infty$) lead to a smaller bias in the discretized payoff $P(X_M)$. To match the time-discretization bias, we thus need to set $M = O(\epsilon^{-1/\alpha})$.

For Euler–Maruyama ($\alpha = 1$ in the weak sense), one thus needs to set $M = O(\varepsilon^{-1})$ to make the bias lower than ε.

Once M is fixed, we run N independent Monte Carlo simulations i.e. N sample paths of length M. By the law of large numbers and central limit theorem, $\mathrm{Var}[\widehat{P}_{\mathrm{MC}}] \propto 1/N$. To get a root-mean-square statistical error ϵ, we need $\frac{\sigma}{\sqrt{N}} \approx \epsilon$ which leads to $N \approx \frac{\sigma^2}{\epsilon^2}$. Hence we must set $N = O(1/\epsilon^2)$ which means that if we require a higher accuracy (smaller ϵ), we are forced to increase the number of simulated paths quadratically in $1/\epsilon$. Note that we can also derive this by using Chebyshev's inequality

$$\mathbb{P}\Big(\big|\widehat{P}_{\mathrm{MC}} - \mathbb{E}_{\mathbb{P}}[P(X_M)]\big| \;\geq\; \varepsilon\Big) \;\leq\; \frac{\mathrm{Var}\big(P(X_M)\big)}{N\,\varepsilon^2},$$

as a fixed confidence level requires $N = O(\varepsilon^{-2})$.

Now, since simulating each path costs $O(M)$ operations, generating all N paths costs $O(N \times M)$. Using the previous results for $\alpha = 1$ (Euler–Maruyama), one thus obtains

$$\boxed{\text{Total Cost} = O\big(\varepsilon^{-2}\big) \;\times\; O\big(\varepsilon^{-1}\big) = O\big(\varepsilon^{-3}\big)}$$

2.4 Improved Complexity with Strong Order

This ε^{-3} scaling can be burdensome for small ε. Higher strong-order schemes can reduce the time-discretization error more aggressively. If one employs a method of strong order r, the pathwise error in X_t scales as M^{-r}, which can be relevant when payoffs depend on the entire path or when coupling paths across different discretization levels for variance reduction. The following theorem formalizes how a higher-order method can lower the exponent in the Monte Carlo cost expression.

Theorem 2.1: Complexity with Strong Order r

Let a numerical SDE solver have strong order $r > 0$. If the bias scales like $O(M^{-r})$, balancing the discretization and sampling errors to achieve a root-mean-square precision ε implies $M = O(\varepsilon^{-1/r})$ and $N = O(\varepsilon^{-2})$. Consequently,

$$\text{Cost} = O(N \times M) = O\big(\varepsilon^{-2}\big) \times O\big(\varepsilon^{-1/r}\big) = O\big(\varepsilon^{-2-1/r}\big).$$

Proof. For a strong order r method, the bias error scales as $O(M^{-r})$, while the variance scales as $O(1/N)$. To achieve a RMSE of ε, we require:

$$\frac{1}{\sqrt{N}} = O(\varepsilon) \quad \text{and} \quad M^{-r} = O(\varepsilon)$$

Solving these gives:

$$N = O(\varepsilon^{-2}) \quad \text{and} \quad M = O(\varepsilon^{-1/r})$$

The total cost is then:

$$\text{Cost}_{\mathrm{MC}} = O(N \cdot M) = O(\varepsilon^{-2} \cdot \varepsilon^{-1/r}) = O(\varepsilon^{-2-1/r})$$

□

High strong order r therefore mitigates the time-discretization overhead, although not only it is quite hard to perform a scheme with large r due to the higher smoothness requirement of the SDE, but also the fundamental sampling limit of $O(\varepsilon^{-2})$ remains. However, many practitioners seek further efficiencies because even an $\varepsilon^{-2.5}$ or $\varepsilon^{-2.2}$ cost can be substantial for large-scale problems. In the next section, we explore Multilevel Monte Carlo (MLMC) to reduce this sampling burden further. MLMC organizes simulations across multiple discretization levels to reduce variance. This strategy lowers the complexity from the naive $O(\varepsilon^{-3})$ to approximately $O(\varepsilon^{-2}\log(1/\varepsilon))$.

3 Multilevel Monte Carlo

Multilevel Monte Carlo (MLMC) alleviates the high computational cost of standard Monte Carlo by exploiting a hierarchy of discretized approximations to the underlying stochastic differential equation as presented in (Giles, 2008) [2].

3.1 Principle

The core idea of MLMC is to construct a sequence of discretizations indexed by $\ell = 0, 1, \ldots, L$, where level ℓ corresponds to an approximation of the SDE with a timestep Δt_ℓ and Δt_L is chosen so that the bias at the finest level does not exceed a target ε. The expectation $\mathbb{E}[P]$ is decomposed into a telescoping sum so that

$$\mathbb{E}\big[P\big(X_L\big)\big] = \mathbb{E}\big[P\big(X_0\big)\big] + \sum_{\ell=1}^{L} \Big[\mathbb{E}\big(P\big(X_\ell\big)\big) - \mathbb{E}\big(P\big(X_{\ell-1}\big)\big)\Big],$$

where each X_ℓ denotes the SDE solution with timestep Δt_ℓ i.e. computed at level ℓ of discretization. Coarse levels (small ℓ) require fewer timesteps but have higher bias, while fine levels (large ℓ) are more accurate but computationally expensive. MLMC balances the bias and variance by optimally allocating computational resources across levels.

Multilevel Monte Carlo estimates each expectation in this telescoping sum with an independent Monte Carlo average of $P\big(X_0\big)$ and $\big(P\big(X_\ell\big) - P\big(X_{\ell-1}\big)\big)$ for $\ell = 1, \ldots, L$. Coupling X_ℓ and $X_{\ell-1}$ through shared Brownian increments can lead to a significant reduction in the variance of $P\big(X_\ell\big) - P\big(X_{\ell-1}\big)$. The method thus refines the classical approach by recognizing that coarse simulations, although less accurate, can offer valuable variance information that reduces the number of expensive fine-scale samples required for a given level of accuracy.

3.2 Complexity

By choosing the number of samples at each level to balance the computational work against the variance contribution, MLMC achieves lower overall cost than a single-level approach. Under typical smoothness and Lipschitz assumptions on the SDE coefficients and payoff, it is sufficient to use $\Delta t_L \approx O(\varepsilon)$ to keep the time-discretization bias near $O(\varepsilon)$. The variances of the differences $P\big(X_\ell\big) - P\big(X_{\ell-1}\big)$ often decay at least as fast as $O(\Delta t_\ell)$, enabling a reduction in the number of fine-level samples compared to the naive scheme of simulating the finest level for all realizations.

In many cases, MLMC reduces the Monte Carlo complexity from $O(\varepsilon^{-3})$ to a regime closer to $O(\varepsilon^{-2}\log(1/\varepsilon))$. This improvement is particularly notable for pricing exotic options where discretization and path-dependence are critical. The method preserves the fundamental flexibility of Monte Carlo but exploits correlated coarse and fine paths to concentrate sampling effort where it matters most. The resulting computational savings can be substantial when aiming for small error tolerances.

Although MLMC was originally devised in a classical setting, its key principles extend naturally to the quantum domain. The hierarchical structure, whereby coarse simulations guide the distribution of samples at finer resolutions, remains valuable even when employing amplitude estimation to accelerate the statistical convergence. Next sections examine how quantum amplitude estimation, combined with the telescoping sum inherent in MLMC, can offer additional speedups in the overall cost. Such hybrid quantum–classical variants of MLMC potentially achieve asymptotic complexities approaching $O(\varepsilon^{-3/2}\,\mathrm{polylog}(1/\varepsilon))$, highlighting a promising approach for large-scale derivative pricing and risk analysis.

4 Quantum Monte Carlo and Q MLMC

4.1 Oracles

In the context of quantum computing, an *oracle* is a black-box quantum operator that encodes information about a function. Oracles play a crucial role in quantum algorithms, serving as the mechanism for

querying data or implementing mathematical operations on quantum states.

In this paper specifically, oracles are used to encode information about stochastic differential equations and the associated payoff functions. First off, the oracle O_P evaluates the payoff function $P(x)$ for a given input x, typically representing the state of a stochastic process at a specific time. Quantum algorithms then use these oracles to compute expectation values efficiently, leveraging the superposition and interference principles of quantum mechanics.

Example 1

An example of an oracle in this paper is the operator:

$$U_P|x\rangle|0\rangle = |x\rangle|P(x)\rangle,$$

where:

- $|x\rangle$ is the quantum state representing the input x.
- $|0\rangle$ is an auxiliary register initialized to a default state.
- $|P(x)\rangle$ is the output state, where the auxiliary register encodes the value of the payoff function $P(x)$.

This oracle is used in quantum amplitude estimation and other algorithms to evaluate and manipulate the payoff function within a quantum computation.

Example 2

Suppose the payoff function $P(x) = \max(x - K, 0)$, representing the payoff of a European call option with strike price K. The oracle U_P acts as:

$$U_P|x\rangle|0\rangle = |x\rangle|\max(x - K, 0)\rangle.$$

In (D. An *et al.*, 2023) [1] these numerical schemes are used to approximate the paths of SDEs for the multilevel Monte Carlo (MLMC) method. The strong order r determines the convergence rate of the numerical approximation, influencing the overall computational complexity.

4.2 Two important results

The following lemmas are foundational results that enable the quantum-accelerated multilevel Monte Carlo (QA-MLMC) algorithm to achieve a quadratic speedup over classical Monte Carlo methods. Their importance is highlighted as follows

Lemma : Quantum Mean Estimation

This lemma provides the essential complexity reduction in estimating the expectation of a random variable $v(A)$, where A is an algorithm (classical or quantum) with bounded variance σ^2. The lemma states that a quantum algorithm can estimate $\mathbb{E}[v(A)]$ with additive error ϵ using

$$O\left(\frac{\sigma}{\epsilon}(\log(\sigma/\epsilon))^{3/2}(\log\log(\sigma/\epsilon))\right)$$

samples.

This near-quadratic speedup in precision, compared to classical Monte Carlo methods, directly reduces the computational complexity of multilevel Monte Carlo methods. Specifically, it ensures that the computational cost scales linearly with $1/\epsilon$ up to polylogarithmic factors, as opposed to quadratically in classical methods. QA-MLMC leverages this lemma to efficiently estimate the telescoping sum of terms that approximate the desired expectation in multilevel schemes.

Lemma : Powering Lemma

The powering lemma enhances the success probability of the Quantum Mean Estimation Lemma. It states that repeating an algorithm $O(\log(1/\delta))$ times and taking the median increases the probability of obtaining an accurate estimate to $1-\delta$ for any $\delta > 0$.

The results presented establish the theoretical foundation of QA-MLMC, enabling a computational complexity of $O(\epsilon^{-1}\text{polylog}(\epsilon^{-1}))$, which is a significant improvement over classical multilevel Monte Carlo methods. More than that, the framework ensures that the telescoping sum used in multilevel schemes is computed accurately and robustly, even in the presence of noise or probabilistic failures. This guarantees that the overall error remains bounded within the prescribed tolerance ϵ.

These lemmas thus justify the feasibility and efficiency of quantum speedup in multilevel Monte Carlo simulations. They are not proven in the paper, but we'll try to give an explanation.

4.3 Proof of the Quantum Mean Estimation Lemma

Let A be an algorithm (classical or quantum) whose output random variable $v(A)$ has variance σ^2. Then the Quantum Mean Estimation Lemma states that there exists a quantum algorithm that estimates $\mathbb{E}[v(A)]$ with additive error ϵ and success probability at least $2/3$ using

$$O\left(\frac{\sigma}{\epsilon}(\log(\sigma/\epsilon))^{3/2}(\log\log(\sigma/\epsilon))\right)$$

oracle calls.

The author refers to Montanaro [3] in which we also find the following important result:

Theorem 4.1: Amplitude Estimation

There is a quantum algorithm called *amplitude estimation* which takes as input:

- One copy of a quantum state ψ,
- A unitary transformation $U = 2\psi\psi - I$,
- A unitary transformation $V = I - 2P$ for some projector P, and
- An integer t,

and outputs $\tilde{a}$, an estimate of $a = \psi P \psi$, such that:

$$|\tilde{a} - a| \leq \frac{2\pi\sqrt{a(1-a)}}{t} + \frac{\pi^2}{t^2},$$

with probability at least $8/\pi^2$. The algorithm uses t applications of U and V.

This theorem is not proved in (Montanaro, 2015) [3] but the proof can be found in (Brassard *et al.*, 2000) [4]. This result shows that we can build an ϵ estimator of a function in $O\left(\frac{1}{\epsilon}\right)$ with probability $8/\pi^2$. Using specific algorithms described in Montanaro's paper we get to the complexity described in the Quantum Mean Estimation Lemma.

4.4 Proof of the Powering Lemma

Let A be an algorithm that estimates μ with additive error ϵ, failing with probability $\gamma < 1/2$. The powering lemma states that repeating A $O(\log(1/\delta))$ times and taking the median of results boosts the success probability to $1-\delta$.

Proof. Let $X_1, X_2, \ldots, X_m$ be m independent outputs of the algorithm A, each satisfying $|X_i - \mu| \leq \epsilon$ with probability $1-\gamma$. The probability that A fails on a single execution is $\gamma < 1/2$.

The median of m independent runs $\{X_i\}$ is used as the final estimate $\tilde{\mu}$. The median ensures that fewer than $\lceil m/2 \rceil$ runs can deviate significantly while maintaining a robust central value.

Using the Chernoff bound, the probability that the median deviates from μ is

$$P(|\tilde{\mu} - \mu| > \epsilon) \leq e^{-m \cdot \kappa},$$

where $\kappa > 0$ depends on $1 - 2\gamma$. Setting $m = O(\log(1/\delta))$ ensures $P(|\tilde{\mu} - \mu| > \epsilon) \leq \delta$.

If $T(A)$ is the runtime of a single execution of A, the runtime of the boosted algorithm is then $m \cdot T(A) = O(T(A) \cdot \log(1/\delta))$. □

4.5 Main Results for Monte Carlo and Quantum Accelerated MLMC

Theorem 4.2: Complexity of Quantum Monte Carlo

Let A be a (classical or quantum) algorithm that generates samples of a random variable with variance σ^2. Using a quantum algorithm, the expectation of the random variable can be approximated to additive error ε with a success probability of at least 0.99. The computational complexity is:

$$O\left(\frac{\sigma}{\varepsilon} \cdot (\log \frac{\sigma}{\varepsilon})^{3/2} (\log\log \frac{\sigma}{\varepsilon})\right)$$

This result leverages the Quantum Amplitude Estimation Lemma, which quadratically reduces the sample complexity compared to classical Monte Carlo methods. While classical Monte Carlo requires $O(\varepsilon^{-2})$ samples, quantum Monte Carlo achieves the same precision with $O(\varepsilon^{-1})$ queries. The additional logarithmic factors account for the technical implementation of amplitude estimation, including confidence amplification and iterative steps in the quantum algorithm.

Theorem 4.3: Complexity of Quantum-accelerated MLMC

Let P denote a random variable, and P_l $(l = 0, 1, \ldots, L)$ a sequence of increasingly accurate approximations to P with corresponding costs C_l and variances V_l. Suppose there exist positive constants $\alpha, \beta = 2\hat{\beta}, \gamma$, with $\alpha \geq \min(\hat{\beta}, \gamma)$, such that:

- $|\mathbb{E}[P_l - P]| = O(2^{-\alpha l})$,
- $V_l = O(2^{-\beta l}) = O(2^{-2\hat{\beta} l})$,
- $C_l = O(2^{\gamma l})$.

Then, for any $\varepsilon < 1/e$, a quantum algorithm can approximate $\mathbb{E}[P]$ to additive error ε with success probability at least 0.99, with computational complexity:

$$\begin{cases} O\left(\varepsilon^{-1} (\log \varepsilon^{-1})^{3/2} (\log\log \varepsilon^{-1})^2\right), & \hat{\beta} > \gamma, \\ O\left(\varepsilon^{-1} (\log \varepsilon^{-1})^{7/2} (\log\log \varepsilon^{-1})^2\right), & \hat{\beta} = \gamma, \\ O\left(\varepsilon^{-1-(\gamma-\hat{\beta})/\alpha} (\log \varepsilon^{-1})^{3/2} (\log\log \varepsilon^{-1})^2\right), & \hat{\beta} < \gamma. \end{cases}$$

where α controls the decay of the bias error across levels, β controls the decay of variance across levels and γ relates to the cost per sample at each level.

Quantum Accelerated MLMC (QA-MLMC) extends the benefits of quantum amplitude estimation to multilevel Monte Carlo. In classical MLMC, the computational cost is governed by balancing variance and bias across multiple levels of approximation. QA-MLMC preserves this hierarchical structure but reduces the complexity of approximating each expectation value $\mathbb{E}[P_l - P_{l-1}]$ from $O(\varepsilon^{-2})$ in the classical case to $O(\varepsilon^{-1})$ with quantum acceleration.

QA-MLMC applies to hierarchical problems requiring variance reduction techniques, such as pricing financial derivatives or high-dimensional integral approximations.

5 MLMC for SDE and Quantum MLMC for SDE

To demonstrate the application of Quantum-Accelerated Multilevel Monte Carlo (QA-MLMC) to solving stochastic differential equations, we consider the example of the Black-Scholes model. This provides a benchmark for understanding the use of QA-MLMC in financial derivatives. As previously seen, the Black-Scholes model describes the price of an asset S_t as a Geometric Brownian Motion:

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

where μ is the drift term (risk-free rate), σ is the volatility of the asset, and W_t is a standard Brownian motion. The price of a financial derivative $V(s,t)$, such as a European call option, satisfies the Black-Scholes PDE i.e.

$$\frac{\partial V}{\partial t} + \frac{\sigma^2 s^2}{2}\frac{\partial^2 V}{\partial s^2} + \mu s \frac{\partial V}{\partial s} = \mu V,$$

with terminal condition $V(s,T) = \psi(s)$, where $\psi(s)$ is the final payoff. By Feynman-Kac theorem, this solution can be expressed as

$$V(s,t) = \mathbb{E}\left[e^{-\mu(T-t)}\psi(S_T) \mid S_t = s\right]$$

5.1 Numerical Approximation via Euler-Maruyama Scheme

To simulate paths of the underlying asset price, we discretize the SDE using the Euler-Maruyama scheme:

$$S_{k+1} = S_k + \mu S_k h + \sigma S_k \Delta W_k,$$

where $\Delta W_k \sim \mathcal{N}(0,h)$ represents increments of the Brownian motion, and h is the time step. The goal is to estimate $\mathbb{E}[\psi(S_T)]$ for the terminal payoff $\psi(S_T) = \max(S_T - K, 0)$, corresponding to a European call option.

5.2 Quantum-acceleration of MLMC

Recall that the classical Multilevel Monte Carlo (MLMC) method employs a telescoping sum:

$$\mathbb{E}[P] = \mathbb{E}[P_0] + \sum_{l=1}^{L} \mathbb{E}[P_l - P_{l-1}],$$

where P_l represents the payoff calculated with increasing discretization accuracy at level l. QA-MLMC enhances this framework by leveraging quantum acceleration in the estimation of each term $\mathbb{E}[P_l - P_{l-1}]$.

Lemma : Corollary 2, p.19

Consider a payoff function P satisfying certain smoothness conditions. QA-MLMC estimates $\mathbb{E}[P]$ up to additive error ε with probability at least 0.99 and achieves the following complexity:

$$\tilde{O}(\varepsilon^{-1}), \quad r > 2,$$

$$\tilde{O}(\varepsilon^{-1/2-1/r-o(1)}), \quad r \leq 2,$$

where r is the strong convergence order of the numerical scheme.

This result highlights the quadratic speedup in precision ε achieved by QA-MLMC over classical MLMC. For payoff functions with globally Lipschitz continuity, the complexity can be further improved:

$$\tilde{O}(\varepsilon^{-1}), \quad r \geq 1$$

$$\tilde{O}(\varepsilon^{-1/r}), \quad r < 1$$

6 Application to European call options

In this section, we build a Quantum Multilevel Monte Carlo (QMLMC) algorithm that combines classical multilevel Monte Carlo methods and quantum amplitude estimation to efficiently price European call options. Below, we provide a comprehensive explanation of the algorithm and its components.

6.1 Quantum Amplitude Estimation Algorithm

Quantum Amplitude Estimation (QAE) is a quantum algorithm designed to estimate expectation values with a quadratic speedup compared to classical Monte Carlo methods. In our implementation, QAE is combined with the Multilevel Monte Carlo (MLMC) framework to compute the price of a European call option efficiently. Below, we summarize the main concepts, key equations, and components of the algorithm.

1°) Black-Scholes Equation: Recall that the price of a European call option in the Black-Scholes model is given by

$$C(S, K, T, r, \sigma) = S\Phi(d_1) - Ke^{-rT}\Phi(d_2), \tag{1}$$

where

$$d_1 = \frac{\ln(S/K) + (r + 0.5\sigma^2)T}{\sigma\sqrt{T}},$$
$$d_2 = d_1 - \sigma\sqrt{T},$$

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

2°) Stock price dynamics: In our algorithm, the underlying asset price S_t follows a geometric Brownian motion modeled as:

$$dS_t = \mu S_t dt + \sigma S_t dW_t, \tag{2}$$

where μ is the drift, σ is the volatility, and dW_t is a Wiener process.

3°) Euler-Maruyama Scheme: To simulate S_t numerically, we use the Euler-Maruyama discretization:

$$S_{t+\Delta t} = S_t \exp\left((\mu - 0.5\sigma^2)\Delta t + \sigma\sqrt{\Delta t}Z\right), \tag{3}$$

where $Z \sim \mathcal{N}(0, 1)$. The Euler-Maruyama scheme is of strong order 0.5, suitable for simulating paths in Monte Carlo methods.

Oracle Design and Phase Flip Criterion

In QAE, we use an oracle to mark states contributing positively to the expectation value. The oracle implements a phase flip based on the criterion $f(x) > 0$, where $f(x)$ is the difference in payoffs between consecutive levels in the MLMC framework:

$$f(x) = P_l(x) - P_{l-1}(x). \tag{4}$$

The oracle acts as follows:

$$O_f|x\rangle|f(x)\rangle = \begin{cases} -|x\rangle|f(x)\rangle, & \text{if } f(x) > 0, \\ |x\rangle|f(x)\rangle, & \text{otherwise.} \end{cases} \tag{5}$$

This marking ensures that Grover iterations amplify the amplitudes of the "good" states, increasing their probability of being measured.

Qubit Usage

Our implementation uses a total of 7 qubits:

- 3 qubits encode the log-normal distribution of the underlying stock price.
- 1 qubit is used for the payoff function (objective qubit).
- 3 ancillary qubits support arithmetic operations and oracle implementation.

Algorithm Workflow

1. **State Preparation:** Prepare a quantum state encoding the log-normal distribution and payoff function.
2. **Oracle Marking:** Apply the oracle to mark states satisfying the criterion $f(x) > 0$.
3. **Amplitude Amplification:** Use Grover iterations to amplify the marked states' amplitudes.

4. **Measurement:** Measure the quantum state and extract the expectation value through post-processing.

5. **MLMC Framework:** Combine the results from different levels using the telescoping sum:

$$\mathbb{E}[P] = \mathbb{E}[P_0] + \sum_{l=1}^{L} \mathbb{E}[P_l - P_{l-1}]. \tag{6}$$

6.2 Results

Log-Normal Distribution

State	Price Range	Probability	Amplitude	Payoff
$\|000\rangle$	$[0, S_1]$	p_0	$\sqrt{p_0}$	$\max(S_0 - K, 0)$
$\|001\rangle$	$[S_1, S_2]$	p_1	$\sqrt{p_1}$	$\max(S_1 - K, 0)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Table 1: Example of discretized log-normal states and payoffs.

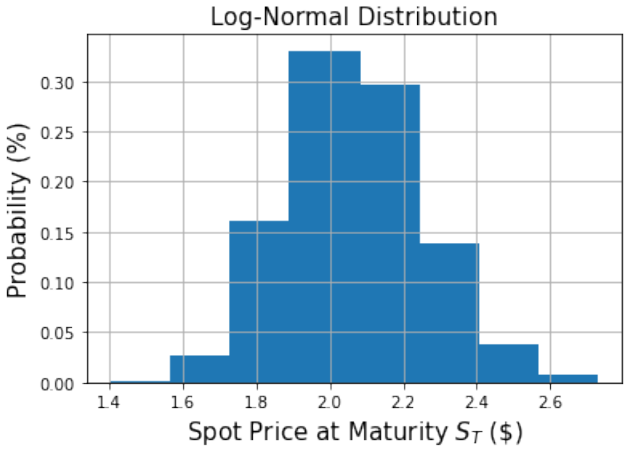

Figure 1: Distribution in 8 qubits for the log return

We fixed the epsilon target at 0.15 because the memory of classic computer is not enough for a more ambitious precision. Let's see the results.

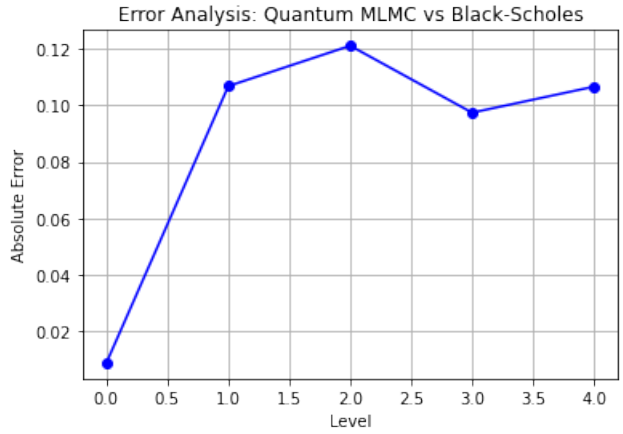

Figure 2: Precision depending of MLMC levels

In the associated notebook we compare the runtime between the classical Monte Carlo Algorithm and the Quantum Algorithm. The Q MLMC takes about $5 \cdot 10^{-5}$ seconds while the classic Monte Carlo takes 10^{-2} seconds which makes sense since the epsilon target we fixed was around 0.1. Indeed we've seen previously that the Q MLMC complexity is around $\tilde{O}(\varepsilon^{-1})$ when the Classic Monte Carlo without any strong order scheme is $O(\varepsilon^{-3})$. However, this algorithm is not applicable in practice without quantum computers since the Quantum Amplitude Estimation Algorithm requires too much memory.

Conclusion

This work demonstrates the potential of quantum-accelerated multilevel Monte Carlo (QA-MLMC) to substantially reduce the computational complexity of SDE simulations in quantitative finance. By leveraging quantum amplitude estimation within the MLMC framework, the overall complexity for achieving an error tolerance of ε is reduced from the classical $O(\varepsilon^{-3})$ to $\tilde{O}(\varepsilon^{-1})$, representing a near-quadratic speedup. This acceleration is critical for pricing complex financial derivatives and performing large-scale simulations, where classical Monte Carlo methods become prohibitively expensive. Furthermore, the hierarchical nature of MLMC, combined with quantum speedup, allows for efficient allocation of computational resources across different discretization levels, optimizing both accuracy and runtime.

While the practical implementation of QA-MLMC remains dependent on the availability of scalable quantum hardware, the theoretical advancements presented highlight the transformative potential of quantum algorithms in computational finance. As quantum technologies spread, QA-MLMC offers a promising pathway toward addressing computational bottlenecks inherent to classical Monte Carlo methods, paving the way for faster and more accurate financial modeling.

Bibliography

[1] D. An, N. Linden, J.-P. Liu, A. Montanaro, C. Shao, and J. Wang. Quantum-accelerated multilevel Monte Carlo methods for stochastic differential equations in mathematical finance, 2023.

[2] M. B. Giles. Multilevel Monte Carlo path simulation. *Operations Research*, 2008.

[3] A. Montanaro. Quantum speedup for Monte Carlo methods. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 2015.

[4] G. Brassard, P. Høyer, M. Mosca, and A. Tapp. Quantum amplitude amplification and estimation. *Quantum Computation and Information*, 2000.