

CONNECTING QUANTUM COMPUTING WITH CLASSICAL STOCHASTIC SIMULATION

Jose Blanchet¹, Mark S. Squillante², Mario Szegedy³, and Guanyang Wang⁴

¹Dept. of Mgmt. Science and Eng., Stanford University, Stanford, CA, USA

²Mathematics of Computation, IBM Research, Yorktown Heights, NY, USA

³Dept. of Computer Science, Rutgers University, Piscataway, NJ, USA

⁴Dept. of Statistics, Rutgers University, Piscataway, NJ, USA

ABSTRACT

This tutorial paper introduces quantum approaches to Monte Carlo computation with applications in computational finance. We outline the basics of quantum computing using Grover’s algorithm for unstructured search to build intuition. We then move slowly to amplitude estimation problems and applications to counting and Monte Carlo integration, again using Grover-type iterations. A hands-on Python/Qiskit implementation illustrates these concepts applied to finance. The paper concludes with a discussion on current challenges in scaling quantum simulation techniques.

1 SETTING THE STAGE

Benioff (1980) proposed a quantum version of a Turing machine and showed that such a device could be described using the equations of quantum mechanics. This was followed by the First Conference on the Physics of Computation, jointly organized by MIT and IBM in the summer of 1981, at which both Benioff and Richard Feynman gave keynote talks on quantum computing. Feynman’s keynote, in particular, addressed simulating physics with computers and introduced the concept of a “quantum simulator”—a machine that, unlike a universal Turing machine, specializes in simulating quantum mechanical phenomena. This talk, subsequently published in (Feynman 1982), is often credited with launching quantum computing as a distinct discipline. It ended with his now-famous remark:

“...nature isn’t classical, dammit, and if you want to make a simulation of nature, you’d better make it quantum mechanical, and by golly it’s a wonderful problem, because it doesn’t look so easy.” – Feynman

Quantum computing has since evolved far beyond the simulation of quantum systems alone.

Why Quantum Computing? Quantum computing has captured wide attention due to its potential to solve certain problems much faster than classical computers. For example, Shor’s algorithm showed that a quantum computer could factor large numbers in polynomial time (Shor 1994; Shor 1997), threatening classical RSA encryption which underpins many systems in the financial industry. While practical applications are still emerging, the promise of exponential or significant speedups is a key motivator.

Overview of Quantum Computing Architectures: Depending on how we classify them, quantum computing hardware falls into two or three primary paradigms:

- **Traditional Gate-based quantum computers:** Gate-based quantum computers use logical quantum gates, quantum analogs of classical logic gates, to run algorithms. Platforms include superconducting quantum bits (e.g., IBM, Google, Rigetti), trapped-ion systems (e.g., IonQ, Quantinuum), cold-atom devices (e.g., QuEra), and photonic devices (e.g., Xanadu, PsiQuantum). These machines are universal, capable in principle of executing any algorithm in the class BQP, the complexity class for quantum polynomial-time computation.
- **Quantum annealers (adiabatic quantum computers):** These are designed to solve optimization problems by exploiting quantum physics to find low-energy states. Notably, D-Wave’s machines

use quantum annealing to find the ground state of an Ising model or equivalently solve a Quadratic Unconstrained Binary Optimization (QUBO) problem (Albash and Lidar 2018). However, the computational power of annealers remains an open research question.

While annealing machines work with a large number of qubits ($> 10^3$) and potentially deliver at least quadratic speedups with additional significant constant-factor advantages, their classification as full quantum computers remains contested. Meanwhile, any quadratic speedup would require a very careful control of error-correction resources because a large error-correction overhead can negate such performance gains. In contrast, experiments with existing gate-based quantum computers showed that artificial problems can be crafted with input-output functionalities that are not simulable with realistic classical resources. In spite of this advance, these systems continue to grapple with decoherence and gate errors—challenges that define the current Noisy Intermediate-Scale Quantum (NISQ) era, characterized by devices with hundreds of quantum bits and gate-error rates between 0.1-1%. The field is now actively pursuing technologies that transcend these NISQ-era limitations to realize the full potential of quantum computation, while quantum-bit counts continue to grow exponentially. For instance, IBM alone exhibited a factor of 10 increase between 2016 and 2019, and nearly a factor of 5 increase between 2020 and 2022.

Quantum Error Correction vs. Quantum Hardware Approaches to Reduce Errors: Quantum states are fragile, and today’s quantum bits are error-prone. Quantum error correction (QEC) encodes a single logical quantum bit into multiple physical quantum bits, detecting and correcting errors to store information reliably. QEC is theoretically necessary for scalable, fault-tolerant quantum computing (Nielsen and Chuang 2010) unless one could build a device with zero error. In fact, at least a year passed between Shor’s algorithm for ideal, error-free quantum bits and the mathematical proof that quantum mechanics, in principle, permits its implementation on imperfect physical quantum bits.

The first step was made by Peter Shor himself, who constructed the first quantum error-correcting code (QECC), the 9-quantum-bit Shor code (independently, a 7-quantum-bit code by Steane was discovered); refer to Shor (1995), Steane (1996). Unlike classical codes, QECCs must correct both bit-flip and phase-flip errors and their combinations. However, a fully fault-tolerant architecture requires not only QECCs but also fault-tolerant gate constructions. To this end, researchers have focused on a subclass of QECCs, called CSS codes named after Calderbank, Shor, and Steane (Calderbank and Shor 1996; Steane 1996), which allows separate correction of bit- and phase-flip errors and conveniently supports transversal Clifford gates. Virtually all practical QECCs in current use are CSS codes, or more generally stabilizer codes, an elegant mathematical formalism developed by Gottesman (1997); see also Calderbank et al. (1998). The celebrated threshold theorem (Aharonov and Ben-Or 1997; Kitaev 1997; Knill et al. 1998) established that, if physical error rates lie below a constant threshold, arbitrarily long computations can be realized with only polylogarithmic overhead in quantum-bit count. At first, due to its complexity, it remained only a proof of concept.

Subsequent development included Kitaev’s surface code (Kitaev 2003) and its several variations as practical QECCs. Bravyi and Kitaev (2005) introduced magic-state distillation, greatly simplifying the result of Aharonov and Ben-Or. Yet QEC continues to raise deep questions both in theory and practice, and it is a hot topic.

QEC overhead. Depending on quantum-bit quality, error correction can increase physical quantum-bit counts by factors of thousands and add significant time overhead. Estimates for tasks like breaking RSA with Shor’s algorithm often require millions of physical quantum bits. Even solving a chemistry problem, which could be the first killer application of quantum computing, might theoretically need 100 logical quantum bits, but if each logical quantum bit needs $\sim 1,000$ physical quantum bits, that is 100,000 physical quantum bits – far beyond the few hundred available today. Consequently, hardware researchers pursue three main strategies each of which have made recent advances:

1. **Improving coherence and control of quantum bits** across established platforms (superconducting circuits, trapped ions, neutral atoms, and so on).

2. **Clever ways of mitigating errors.** IBM has developed sophisticated error mitigation strategies (Kandala et al. 2019) including zero-noise extrapolation (ZNE) and readout error mitigation that reduce the impact of noise without full error correction. Additionally, emerging error correction techniques focus on detecting and correcting only the most frequent error types rather than achieving full fault tolerance, offering a practical intermediate approach for current hardware.

3. **Exploring novel architectures.** Topological quantum computing—pursued by Microsoft and several universities—aims to encode information in non-Abelian anyons or other exotic states with intrinsic error protection. If realized, these quantum bits could exhibit dramatically lower error rates. Nevertheless, formidable technical and materials challenges remain before topological quantum bits can become practical, if ever.

Incremental progress continues: the fidelity of quantum bits improve each year, and companies have roadmaps for devices with a thousand quantum bits, but significant scale-up is still required. Experts in the industry estimate a 5-20 year timeframe for practically useful fault-tolerant quantum computers.

Current Quantum-Solvable Problems (Focus on Quantum Versions of Monte Carlo): What can quantum computers currently solve, or are expected to solve soon? We are currently in the NISQ era, which means we must contend with the fact that quantum computing is inherently noisy—and in some approaches, even attempt to utilize this randomness constructively.

Algorithms like the Variational Quantum Eigensolver (VQE) due to Peruzzo et al. (2014) and the Quantum Approximate Optimization Algorithm (QAOA) due to Farhi et al. (2014) are heuristic hybrid approaches that combine quantum resources with classical feedback, aiming to achieve quantum advantage. However, despite these hybrid classical-quantum approaches, practical quantum advantage remains elusive. Nevertheless, theoretical results provide clear direction on several domains where quantum computing should excel once hardware capabilities mature sufficiently.

One such area is Monte Carlo simulation and integration, which is ubiquitous in science and finance. Quantum approaches to Monte Carlo methods aim to speed up the estimation of expected values, probabilities, and other statistical outputs. A prominent result by Montanaro (2015) showed that quantum algorithms can quadratically speed up Monte Carlo methods under certain conditions. Specifically, if a classical Monte Carlo would take N samples to achieve a certain error (uncertainty) ϵ (often $N \sim 1/\epsilon^2$ for Monte Carlo), a quantum computer could in principle achieve a similar error with on the order of \sqrt{N} samples (scaling as $1/\epsilon$) – an almost quadratic speedup.

For example, in finance, Monte Carlo is used to price derivatives or estimate risk metrics. Quantum Amplitude Estimation (QAE) has been proposed to price financial derivatives with fewer samples than classical Monte Carlo due to Rebentrost et al. (2018). IBM researchers Woerner and Egger (2019) demonstrated how QAE can compute risk measures like Value at Risk with far fewer simulations than a classical approach. In general, further advances in quantum approaches to Monte Carlo methods should be welcome because many industries (e.g., insurance, banking, engineering) have expensive Monte Carlo workloads.

It is important to note, though, that these advantages of quantum approaches to Monte Carlo assume we have quantum hardware capable of running the required circuits with low error. In practice, current hardware is not there yet – for instance, QAE algorithms require either deep circuits (with Quantum Fourier Transforms) or multiple circuit repetitions that are challenging on NISQ devices. Researchers are exploring NISQ-friendly variants (like iterative amplitude estimation, which avoids some deep quantum operations) to test these ideas on current machines. So while theoretical speedups exist for Monte Carlo and other problems, demonstrating them experimentally will likely require further advances in hardware or error mitigation.

2 BASIC ELEMENTS OF QUANTUM COMPUTING

Before diving into the quantum speedups of Monte Carlo, let us review the basic elements of quantum computing. We will then apply these notions to explain Grover’s search algorithm, which plays a key role

in the quantum speedup of Monte Carlo methods. We start with the definition of two key concepts: qubits and superpositions. Our presentation here will be brief, but we refer the reader to Nielsen and Chuang (2010), De Wolf (2019), Lipton and Regan (2021) for additional details on quantum computing.

We shall use standard quantum notation throughout this paper, which includes representing quantum states (which, as we shall explain, are simply vectors) with the *bra-ket* notation introduced by Paul Dirac. More specifically, a quantum state is represented as a column vector $|\cdot\rangle$ where different variables inside the *ket* indicate different quantum states; for example, $|s\rangle$ and $|s'\rangle$ represent the (column) vector states s and s' of a quantum system. The corresponding *bra*, represented as $\langle\cdot|$, is the conjugate transpose of the ket (and therefore a row vector); for example, $\langle s|$ and $\langle s'|$ represent the conjugate transpose of the vector states s and s' of a quantum system. Finally, $\langle s|s'\rangle$ represents the inner (scalar) product formed between the two quantum states.

Qubits and Superposition. The basic unit of information in classical computing is the binary bit which takes on values of either 0 or 1. In quantum computing, the basic unit of information is the quantum bit, or *qubit*, which can exist in a *superposition* of the basis states $|0\rangle = (1, 0)^\top$ and $|1\rangle = (0, 1)^\top$ (i.e., can exist in the multiple basis states simultaneously) with complex probability amplitudes. A general (pure) qubit state is written as

$$|s\rangle = \alpha|0\rangle + \beta|1\rangle,$$

where $\alpha, \beta \in \mathbb{C}$ satisfy $|\alpha|^2 + |\beta|^2 = 1$. Here $|0\rangle$ and $|1\rangle$ correspond to the classical bits 0 and 1.

Probability amplitudes in quantum mechanics can interfere constructively (amplification) or destructively (cancellation), as they can be positive, negative, or complex numbers. This quantum interference is key to quantum computational speedups. When measuring the qubit $|s\rangle$, it collapses to state $|0\rangle$ with probability $|\alpha|^2$ or to state $|1\rangle$ with probability $|\beta|^2$, adhering to the normalization condition $|\alpha|^2 + |\beta|^2 = 1$.

More generally, a quantum state is represented as a unit vector in the space of multiple qubits. An n -qubit state is a unit vector in $(\mathbb{C}^2)^{\otimes n}$, where \otimes denotes the tensor product. The standard basis, B_n , of this space has 2^n elements: $B_n = \{|s\rangle \mid s \in \{0, 1\}^n\}$.

For example, the standard basis for a 2-qubit state has $2^2 = 4$ elements: $|00\rangle = |0\rangle \otimes |0\rangle = (1, 0, 0, 0)^\top$, $|01\rangle = |0\rangle \otimes |1\rangle = (0, 1, 0, 0)^\top$, $|10\rangle = |1\rangle \otimes |0\rangle = (0, 0, 1, 0)^\top$, $|11\rangle = |1\rangle \otimes |1\rangle = (0, 0, 0, 1)^\top$.

Operations and Computation. Quantum computing involves operations on the quantum state of n qubits. The state is a vector in a 2^n -dimensional complex Hilbert space (the complex analog of a Euclidean space equipped with an inner product). Quantum operations correspond to multiplying this state vector by $2^n \times 2^n$ unitary matrices. These operations are represented in quantum circuits as quantum gates acting on qubits, with the circuit complexity of a quantum algorithm defined by the number of gates and the circuit depth. The exponentially large Hilbert space in which quantum computations are performed creates the potential for significant speedup over classical algorithms executed on digital computers.

A valid quantum operation applied to a qubit state is a unitary linear transformation (or matrix). Recall that a matrix A is unitary if $AA^\dagger = I$, where $A^\dagger = \bar{A}^\top$ (i.e., the transpose of the complex conjugates of each entry). Equivalently, a linear operator is unitary if it preserves lengths and inner products, so unitary operators can be understood as ‘quantum rotations’. Of course, $AA^\dagger = I$ implies that any such A is reversible. As three important examples, for single qubits, we have the so-called Pauli-X, Pauli-Y and Pauli-Z operators respectively represented by the following three matrices

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \quad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix},$$

where $i = \sqrt{-1}$. They happen to be not only unitary, but also Hermitian (i.e., $A = A^\dagger$).

The matrices X , Y and Z , together with the identity matrix I , form a basis for all 2×2 Hermitian matrices. Every unitary transformation U on a single qubit can be decomposed into products of rotation matrices of the form $\exp(-i\theta Y/2)$ and $\exp(-i\phi Z/2)$ for $\theta \in [0, \pi]$ and $\phi \in [0, 2\pi]$, multiplied by a possible global phase factor: $U = e^{i\alpha} \exp(-i\beta Z/2) \exp(-i\gamma Y/2) \exp(-i\delta Z/2)$. For example, the Hadamard operator or gate,

represented by the matrix $H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$, can be expressed as $H = e^{-3i\pi/2} \exp(-iY\pi/4) \exp(-iZ\pi/2)$ and it satisfies

$$|s\rangle := H|0\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}}.$$

For a qubit initially at state $|0\rangle$, after applying the Hadamard gate, the qubit now has a 50% chance of being either 0 or 1 when measured. The state is neither 0 nor 1 until the measurement, but in a coherent superposition of both 0 and 1.

The k -qubit quantum gates (typically $k = 1, 2$) are unitary operators acting on the k -qubit Hilbert space with dimension 2^k . The reader is referred to Barenco et al. (1995) for more details on elementary quantum gates.

A sequence of quantum gates forms a quantum circuit, analogous to a classical logic circuit. A major aspect of quantum algorithm design is determining how exponentially large unitary matrices can be decomposed into a polynomial-length sequence of gate operations, when possible. The typical quantum algorithm has the following structure: *Input Preparation* \rightarrow *Quantum Operations* \rightarrow *Measurement*. Given that measurement in quantum computing is probabilistic, quantum algorithms often require multiple runs to obtain reliable statistics, and therefore the number of required runs plays an important role in the overall computational complexity of a given quantum algorithm. In addition, the number of gates and the depth of the circuit define the circuit complexity of the quantum algorithm, which characterizes its implementation cost on quantum hardware.

Entanglement. An important property in the setting of a quantum state with multiple qubits is *entanglement*, which refers to a form of quantum correlations between different qubits such that the measurements of the state become correlated in ways that do not have a classical counterpart. To see this, one can reason as follows. Pure states, such as the pair $(|0\rangle, |1\rangle)$, correspond to an arbitrary orthonormal basis. Each such base can be interpreted as a measuring setting. Physically, a basis can correspond to the orientation of a polarizer.

As noted above, an n -qubit quantum state can be expressed as the tensor product of n single-qubit states. On the other hand, it is possible for the n qubits to be in a quantum state that cannot be expressed as the tensor product of n single-qubit states. These states, which cannot be expressed as a tensor product of its single-qubit states, are called entangled states. One simple well-known example is the two-qubit quantum state given by $(|00\rangle + |11\rangle)/\sqrt{2}$, which is an instance of the quantum states considered in the classical paper by Einstein et al. (1935). We observe from this simple example that, if measurement of the first qubit reveals $|0\rangle$, then the two-qubit quantum state collapses to $|00\rangle$, and similarly for a measurement of $|1\rangle$ collapsing to $|11\rangle$. Hence, measurement of the first qubit immediately reveals the value of the second qubit which has not been measured. This property of entanglement is one of the most important properties in creating the possibility of quantum algorithms that are more powerful than their classical counterparts.

To better understand the property of entanglement and how it arises, consider an n_1 -qubit state $|s_1\rangle$ which can be expressed by 2^{n_1} complex numbers whose squared moduli sum to 1 and an n_2 -qubit state $|s_2\rangle$ which can be expressed by 2^{n_2} complex numbers whose squared moduli sum to 1. The tensor product $|s_1 s_2\rangle = |s_1\rangle \otimes |s_2\rangle$ of these two quantum states, while it could take $2^{n_1+n_2}$ outcomes when measured, it depends on $2^{n_1} + 2^{n_2}$ complex numbers. However, an arbitrary quantum state in the joint space of the n_1 -qubit state $|s_1\rangle$ and the n_2 -qubit state $|s_2\rangle$ can be expressed by $2^{n_1+n_2}$ complex numbers whose squared moduli sum to 1. The state $|s_1 s_2\rangle$ lives in a $2^{n_1+n_2}$ -dimensional Hilbert space, where those quantum states that cannot be expressed as a tensor product of the two quantum states $|s_1 s_2\rangle$ are entangled states. Entanglement therefore makes it possible to create a complete 2^n -dimensional Hilbert space in which to perform our computations while using only n qubits. The reader may wonder if in the end it is possible to define a classical probability model on the 2^n -dimensional that is consistent with quantum measurements. The answer is *no*. This is precisely the implication of the thought experiment due to Clauser, Horne,

Shimony, and Holt, leading to the so-called CHSH inequality which is part of a larger set of inequalities first derived by Bell and thus also known as the Bell inequality; see, e.g., Nielsen and Chuang (2010).

Grover's Algorithm: A Basic Example – Unstructured Search. One of the most important quantum algorithms is Grover's search algorithm which tackles the problem of finding a specific marked element ω from among a collection of N elements labeled $1, \dots, N$ (Grover 1996; Grover 1997). Each element is associated with an element of the chosen basis. Grover's algorithm finds the marked element ω with high probability using $O(\sqrt{N})$ well-chosen operations (to be discussed). This running time is known to be of optimal order (Bennett et al. 1997), with even the constant factor being optimal (Zalka 1999), in stark contrast to the best possible classical algorithm, which requires at least $\Omega(N)$ operations in the worst case to find ω with high probability. Hence, the computational complexity of Grover's algorithm provides a quadratic speedup over that of the best classical algorithm for solving the general search problem.

The algorithm begins with the uniform superposition of all basis states on N qubits. This is done by applying the so-called Hadamard gate to each qubit. Call this state $|s\rangle$, with the element of interest $|\omega\rangle$. We can always write

$$|s\rangle = \frac{1}{\sqrt{N}} \sum_{x=1}^N |x\rangle = \frac{1}{\sqrt{N}} |\omega\rangle + \frac{\sqrt{N-1}}{\sqrt{N}} |s'\rangle,$$

where $|s'\rangle = (N-1)^{-1/2} \sum_{x \neq \omega} |x\rangle$ is orthogonal to $|\omega\rangle$ and it represents the uniform superposition over all the unmarked elements. Note that if one were to measure the state $|s\rangle$ immediately after initialization, then the probability of observing the marked element ω is given by $(1/\sqrt{N})^2 = 1/N$, which matches the probability in a classical search.

Grover's algorithm dramatically increases this probability after a sequence of quantum operations. The algorithm can be understood by drawing a two-dimensional picture on the plane generated by the states $|s\rangle$ and $|\omega\rangle$. We place the horizontal axis on the line generated by $|\omega\rangle$ as shown in Figure 1 below. The Grover operator G is the composition of two linear operators, O and D , the first of which computes the reflection in the $|s\rangle$ - $|\omega\rangle$ plane with respect to the $|\omega\rangle$ -axis. More precisely,

$$O = 2|\omega\rangle\langle\omega| - I.$$

Note that if $z = \alpha|\omega\rangle + \beta|s'\rangle$ then $Oz = \alpha|\omega\rangle - \beta|s'\rangle$, which verifies the reflection property with respect to the $|\omega\rangle$ axis. Note that this is just a sign flip in s' .

The second operator computes a reflection with respect to the axis generated by $|s\rangle$ and then it flips the sign of the resulting vector, namely

$$D = -(2|s\rangle\langle s| - I).$$

Both of these operators are valid unitary operators (and thus valid quantum operations). Grover's operator therefore takes the form $G = DO$.

Let $\theta \in [0, \pi/2]$ be the angle between $|s\rangle$ and $|\omega\rangle$. Note that we have $\langle s|\omega\rangle = \cos(\theta) = 1/N^{1/2}$. Upon defining $\phi = \pi/2 - \theta$, we obtain $\sin(\phi) = 1/N^{1/2}$, which implies both that $\phi \approx 1/N^{1/2}$ and that

$$|s\rangle = \sin(\phi)|\omega\rangle + \cos(\phi)|s'\rangle.$$

Each application $r = G|s\rangle$ results in a state r that is rotated by an angle $\pi - 2\theta = \pi - 2(\pi/2 - \phi) = 2\phi$. Then, G^m yields a state that is rotated by an angle of $2m\phi$ units and therefore

$$G^m|s\rangle = \sin(2m\phi)|\omega\rangle + \cos(2m\phi)|s'\rangle.$$

Consequently, we want the resulting angle, namely $2m\phi$, to be as close as possible to $\pi/2$ so that $\sin(2\pi m)^2 \approx 1$, which leads to $m \approx \pi N^{1/2}/4$ for N large. A measurement of the resulting state should then yield ω with high probability; see Brassard and Høyer (1997), Brassard et al. (2002).

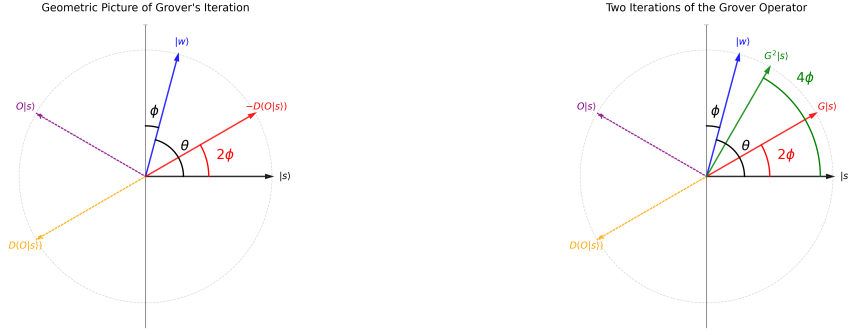


Figure 1: Left: Geometric illustration of Grover's iteration showing the relationship between the state vectors and angles θ , ϕ , and 2ϕ . Right: Geometric illustration of multiple iterations of Grover's operator.

3 FROM GROVER TO QUANTUM SPEEDUP OF MONTE CARLO METHODS

The central technical idea in Grover's algorithm is what is now known as *amplitude amplification*, which uses quantum interference to boost the probability amplitude of a target state at a rate that exceeds what classical methods can achieve. Although originally introduced by Grover for unstructured search, this technique was later formalized and generalized by Brassard et al., making it applicable to a much wider class of computational problems. Following the path of Brassard et al. (2002) and Montanaro (2015), we will take a brief journey from Grover's search algorithm to quantum approaches to Monte Carlo methods. Figure 2 (Left) shows a schematic diagram of this progression. The figure traces a progression of quantum algorithms. The horizontal axis shows how broad the underlying problem class is: a point farther to the right means you are answering a richer statistical query, moving from a binary search (Grover's algorithm) to counting marked items, and finally to estimating an expectation as in Monte Carlo. The vertical axis captures the algorithmic sophistication required to obtain the quadratic speedup that these methods share: Grover's search is the simplest routine, quantum counting builds on it, and the "quantum speedup of Monte Carlo" is the most general and technically involved. A summary table of the main algorithms is provided in Table 1.

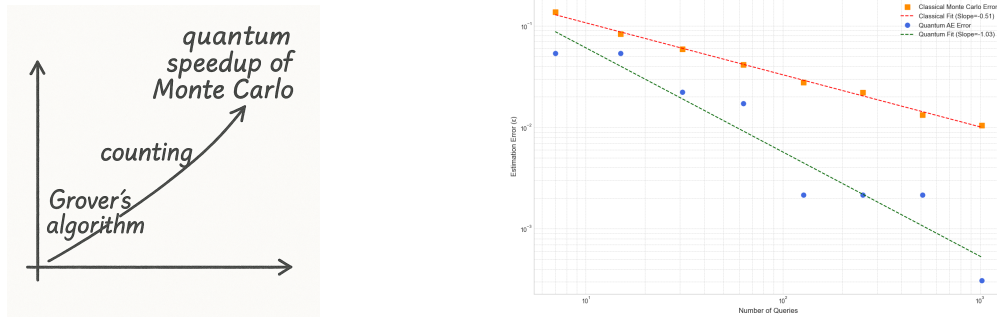


Figure 2: Left: From Grover's algorithm to quantum speedup of Monte Carlo methods. Right: Illustration of quadratic speedup.

3.1 From Grover to Counting: A First Approach to Amplitude Estimation

Consider the problem of estimating the number of elements k that satisfy some property. This problem is equivalent to computing $p = \frac{k}{N}$, which we will do entirely by Grover rotations and simple measurements.

Recall from Section 2 that we set $\sin \phi = \sqrt{\frac{k}{N}} = \sqrt{p}$ and $\cos \phi = \sqrt{1 - p}$.

Table 1: Main algorithms and their classical analogs.

Quantum Algorithm	Classical Analog	Description of Uses
Grover’s Algorithm	Unstructured Search	Finds an item in an unsorted collection.
Quantum Amplitude Estimation (QAE)	Bernoulli Parameter Estimation	Estimates a probability using Grover-style rotations to amplify the “success” state.
Quantum-Accelerated Monte Carlo	General Monte Carlo Methods	Extends QAE to estimate the expected value of a general function, $\mathbb{E}[f(X)]$.

There are two approaches to this problem. One approach is to estimate the phase φ and then estimate p , which is the approach taken by Brassard et al. (2002). The method we pursue here is based on the approach of Aaronson and Rall (2020). Once again, we assume that the uniform superposition has been applied. Then, we assume that k distinguished elements are marked in the basis; as before, they altogether are represented by ω , which leads to

$$|s\rangle = \frac{1}{\sqrt{N}} \sum_{x=1}^N |x\rangle = \sin \varphi |\omega\rangle + \cos \varphi |s'\rangle.$$

Since each Grover iterate $G = DO$ rotates this 2-dimensional plane by 2φ , we obtain just as in the Grover picture that $G^m |s\rangle = \sin((2m+1)\varphi) |\omega\rangle + \cos((2m+1)\varphi) |s'\rangle$. Measuring $G^m |s\rangle$ in the basis $|x\rangle$, $x = 1, \dots, N$, thus yields the marked item ω with probability

$$p_m = \mathbb{P}(\text{“marked”}) = \sin^2((2m+1)\varphi).$$

We can repeat this prepare-Grover-iterate-measure loop R times (obtaining R i.i.d. replications) and therefore compute an empirical estimate $\hat{p}_m(R) = p_m + \eta_{m,R}$, where $\eta_{m,R}$ represents the error captured by the Central Limit Theorem and thus $\eta_{m,R} = O_p(1/R^{1/2})$. We then invert the rotation classically, rendering

$$\hat{\varphi} = \frac{1}{2m+1} \arcsin(\sqrt{p_m + \eta_{m,R}}) \quad \text{and} \quad \hat{p} = \sin^2 \hat{\varphi}.$$

We now can perform an expansion using elementary calculus to verify that, if $m = O(1/\varepsilon)$ and R grows slowly (say $\log(1/\varepsilon)$), we can conclude that $|\hat{p} - p| \leq \varepsilon$. Hence, we recover the same quadratic speedup observed earlier directly as a consequence of Grover’s algorithm and classical statistical measurements.

To sketch some of the details, dropping the subindex in $\eta_{m,R}$, let $\theta = (2m+1)\varphi$ where in contrast with Grover we want to choose θ close to $\pi/4$. We aim to make *small changes* in p show up as *big, easily measured* changes. Thus, we want to maximize $\frac{d}{d\theta} \sin^2 \theta = \sin(2\theta)$, which leads to the target $\theta \approx \pi/4$. Then, $\arcsin(\sqrt{p_m + \eta}) \approx \theta + \eta + O(\eta^2)$ and $\hat{\varphi} = \varphi + O(\eta/m)$. From this we can propagate the error into \hat{p} to conclude $|p - \hat{p}| = O(\eta/m)$, thus deducing that $m = O(1/\varepsilon)$ and R essentially constant does the trick.

Figure 2 (Right) illustrates the quadratic speedup achieved by quantum amplitude estimation. We estimate the success parameter p of a Bernoulli random variable using both quantum amplitude estimation (blue) and classical Monte Carlo (orange) via Qiskit. We additionally fitted a regression line of $\log(\text{error})$ versus $\log(\text{cost})$ for both methods. The fitted curve clearly indicates that amplitude estimation achieves a $1/\varepsilon$ scaling, offering a quadratic speedup compared to the $1/\varepsilon^2$ scaling of classical Monte Carlo.

3.2 Amplitude Estimation for a General State-Preparation Operator

Suppose we have a unitary A that prepares the state $|\psi\rangle = A|0\rangle = \sqrt{p} |\omega\rangle + \sqrt{1-p} |\psi'\rangle$, $p \in [0, 1]$, where $|\omega\rangle$ is the “good” state and $|\psi'\rangle$ spans the orthogonal complement so that $\{|\omega\rangle, |\psi'\rangle\}$ is an orthonormal basis of the two-dimensional subspace of interest. As before, we set $\sin \varphi = \sqrt{p}$ and $\cos \varphi = \sqrt{1-p}$. Define the two reflections as

$$O := I - 2 |\omega\rangle\langle\omega| \quad (\text{oracle on } |\omega\rangle),$$

$$D_\psi := 2 |\psi\rangle\langle\psi| - I = A (2 |0\rangle\langle 0| - I) A^\dagger \quad (\text{diffusion about } |\psi\rangle).$$

Their product $G = D_\psi O$ rotates the $\{|\omega\rangle, |\psi'\rangle\}$ plane by 2ϕ . In particular,

$$G^m |\psi\rangle = \sin((2m+1)\phi) |\omega\rangle + \cos((2m+1)\phi) |\psi'\rangle.$$

A measurement in the $\{|\omega\rangle, |\psi'\rangle\}$ basis then yields a “good” state with probability $p_m = \sin^2((2m+1)\phi)$. By repeating the prepare-rotate-measure sequence R times, one obtains an empirical $\hat{p}_m \approx p_m$. Inverting classically then renders

$$\hat{\phi} = \frac{1}{2m+1} \arcsin(\sqrt{\hat{p}_m}) \quad \text{and} \quad \hat{p} = \sin^2 \hat{\phi}.$$

Choosing $m = O(1/\varepsilon)$ (and constant R) guarantees $|\hat{p} - p| \leq \varepsilon$ using $O(m) = O(1/\varepsilon)$ applications of G , thus generalizing the quadratic-speedup amplitude estimation to any state-preparation unitary A .

3.3 Quantum Monte Carlo via Grover-Style Amplitude Estimation

Now we are ready to estimate $\mathbb{E}[f(X)] = \sum_{x \in \{0,1\}^{Nd}} q(x) f(x) = p$, where $f(x) \in [0,1]$. This can be seen as an approximation to the expectation of a Lipschitz f function in $[0,1]^d$ assuming that $f(x) \in [0,1]$. We will apply a very similar procedure as above.

1. State preparation operator \mathcal{A} . Define \mathcal{A} by

$$\mathcal{A} |0^{Nd+1}\rangle = \sum_x \sqrt{q(x)} |x\rangle_{Nd} \left(\sqrt{f(x)} |1\rangle + \sqrt{1-f(x)} |0\rangle \right) =: |\psi\rangle,$$

so that the final ancilla qubit has “|1>” with overall probability $p = \mathbb{E}[f(X)]$. Here, the notation 0^{Nd+1} indicates the dimension of the vector 0 and the notation $|x\rangle_{Nd}$ indicates a vector in the Nd quantum register. Note that, if $q(x)$ is uniform, we can use the Hadamard gate as in previous examples. Moreover, if $f(x)$ can be evaluated easily on a classical computer, then the unitary \mathcal{A} can be easily implemented as well. A specific example is given in Section 5.3.

2. Grover-geometry. Set

$$\sin \phi = \sqrt{p}, \quad \cos \phi = \sqrt{1-p}, \quad |\psi'\rangle = \frac{1}{\sqrt{1-p}} \sum_x \sqrt{q(x)(1-f(x))} |x\rangle |0\rangle.$$

Then, in the two-dimensional subspace spanned by $|\omega\rangle := \sum_x \sqrt{q(x)f(x)} |x\rangle |1\rangle$ and $|\psi'\rangle$, we have $|\psi\rangle = \sin \phi |\omega\rangle + \cos \phi |\psi'\rangle$.

3. Reflections and rotation. Define the “good” oracle as $O := I - 2 |\omega\rangle\langle\omega|$, and the diffusion about $|\psi\rangle$ as

$$D_\psi := 2 |\psi\rangle\langle\psi| - I = A (2 |0\rangle\langle 0| - I) A^\dagger.$$

Then their product $G = D_\psi O$ rotates that 2-dimensional plane by 2ϕ :

$$G^m |\psi\rangle = \sin((2m+1)\phi) |\omega\rangle + \cos((2m+1)\phi) |\psi'\rangle.$$

4. Re-prepare & measure. For each run, we have

$$|0\rangle \xrightarrow{\mathcal{A}} |\psi\rangle \xrightarrow{G^m} G^m |\psi\rangle \xrightarrow{\text{measure ancilla}} \text{“good” with probability } p_m = \sin^2((2m+1)\phi).$$

Repeating this $R = O(1)$ times yields $\hat{p}_m \approx p_m$, from which we obtain $\hat{\phi} = \frac{1}{2m+1} \arcsin \sqrt{\hat{p}_m}$ and $\hat{p} = \sin^2 \hat{\phi}$. Choosing $m = O(1/\varepsilon)$ renders $|\hat{p} - p| \leq \varepsilon$ with only $O(1/\varepsilon)$ total calls to G , i.e., a quadratic speedup.

Montanaro (2015) extends QAE to the general Monte Carlo problem. Assuming f has finite second moment (with a known upper bound) under p , Montanaro’s algorithm proceeds by applying the idea to dyadic intervals. Precisely, it suffices to assume that $f \geq 0$ (you can write $f = f^+ - f^-$). Then, simply estimate using the above algorithm $\mathbb{E}[f(X)/2^{l+1} I(f(X) \in [2^l, 2^{l+1}))]$ for $l = O(\log(1/\varepsilon))$. This strategy achieves $\tilde{O}(1/\varepsilon)$ cost to estimate $\mathbb{E}[f(X)]$ with an error up to ε . Here, using standard notation, $\tilde{O}(\cdot)$ is the same as $O(\cdot)$ but simply ignoring logarithmic factors.

3.4 Further Developments

There have been many further developments since Montanaro (2015), some of which we mention here.

Algorithm improvements: Montanaro’s algorithm has two limitations. First, it requires a predetermined upper bound on the variance. Second, the error bound includes additional polylogarithmic factors in the parameter ε . Both limitations are resolved in the recent work of Hamoudi (2021) and Kothari and O’Donnell (2023). In particular, Kothari and O’Donnell’s algorithm is state-of-the-art. Consider a random variable \mathbf{y} with a finite mean μ and finite variance σ^2 . Suppose a user wants to estimate μ and has access to the code that generates samples of \mathbf{y} . Kothari and O’Donnell’s algorithm requires $O(n)$ executions of this code and produces an estimator $\hat{\mu}$ that, with high probability, satisfies $|\hat{\mu} - \mathbb{E}[\mathbf{y}]| \leq \sigma/n$.

The mean estimation problem has been generalized in other directions. Cornelissen et al. (2022) consider the multivariate mean estimation problem and present near-optimal quantum algorithms that explicitly depend on the accuracy level ε , the dimensionality d , and the covariance matrix Σ . When the random variable has infinite variance but a finite $(1 + \delta)$ -th moment, Blanchet et al. (2024) propose a quantum algorithm similar to Montanaro (2015) that also achieves near-quadratic speedup.

Practical algorithms on near-term devices: Nearly all quantum approaches to Monte Carlo methods rely on QAE. The original QAE method employs a quantum phase estimation subroutine due to Brassard et al. (2002) that achieves optimal scaling in theory. However, it requires many controlled amplification operations followed by a quantum Fourier transform, resulting in a circuit that is both deep and resource-intensive. Consequently, this approach is often impractical for near-term quantum devices.

There have been several new algorithms that aim to perform QAE without phase estimation. Suzuki et al. (2020) employ a maximum likelihood estimator (MLE) to estimate the mean from the combined measurement data. Similarly, Grinko et al. (2021) introduce an iterative algorithm where MLE is applied at each iteration to update the estimate. Both algorithms seem to achieve better empirical performance than the original QAE. Meanwhile, Aaronson and Rall (2020) adopt a sequence of hypothesis tests, based on amplitude amplification, to progressively narrow down the possible range of the true amplitude.

4 FINANCIAL EXAMPLE USING QUANTUM APPROACH TO MONTE CARLO

Quantum versions of Monte Carlo offer a promising approach for financial applications, such as pricing basket options whose payoffs depend on high-dimensional correlated assets. Classical Monte Carlo can suffer from slow convergence in such settings. The quadratic speedup of quantum approaches to Monte Carlo makes it possible to achieve comparable accuracy with far fewer samples.

Many problems in finance involve computing a quantity of the form $\mathbb{E}_Q[U(X)]$, where U represents a utility or payoff function. The random element X has distribution Q , and X may be a random variable or a random vector, or a continuous-time stochastic process (i.e., a path). In this section, we illustrate the concepts through a series of examples related to option pricing, with increasing levels of complexity. In the subsequent section, we present a case study on applying quantum algorithms to Credit Risk Analysis using Qiskit. For more extensive discussions, we refer the reader to survey papers such as those by Orús et al. (2019) and Herman et al. (2023).

Analytically solvable: Consider a simple European call option, where the underlying asset price S_t at time t evolves as $S_t = S_0 \exp(\sigma W_t + (r - \sigma^2/2)t)$, with W_t denoting a Brownian motion and σ, r being fixed constants. The payoff function is defined by $U(x) := \max\{0, x - K\}$ for some fixed strike price K , and the option price is defined as $\mathbb{E}[U(S_T)]$. In this setting, the Black–Scholes formula provides a closed-form expression for the option price.

‘Vanilla’ Monte Carlo: When the payoff function is more complex, an analytic solution is often unavailable. For instance, the European power option has a payoff of the form $U(x) = \max\{0, x^p - K\}$. For general values of p , this expectation is typically estimated using standard Monte Carlo methods. In such cases, quantum approaches to Monte Carlo (Montanaro 2015) can provide a quadratic speedup over

classical Monte Carlo approaches. Similarly, quantum approaches to Monte Carlo can be directly used to speed up the pricing of basket options.

Multilevel Monte Carlo: Sometimes the underlying process also has to be approximated. For example, the Local Volatility model assumes the underlying price satisfies: $dS_t = \mu S_t dt + \sigma(S_t, t) S_t dW_t$. In practice, the underlying process must be approximated using a numerical discretization method, such as the Euler–Maruyama scheme. In this setting, the estimation accuracy is influenced by both the discretization error (bias) and the number of Monte Carlo samples (variance). Classically, the multilevel Monte Carlo method (Giles 2015; Rhee and Glynn 2015) is widely used to optimally balance bias and variance, allowing one to reach the desired level of precision with minimal computational cost. Similarly, quantum-accelerated multilevel Monte Carlo methods (An et al. 2021) have been proposed to reduce the required number of samples. The degree of speedup ranges from quadratic to sub-quadratic, depending on the structure of the problem.

Monte Carlo inside Monte Carlo (Nested Monte Carlo): Many financial derivatives, such as American options, permit early exercise. As a result, their valuation depends not only on the current underlying price but also on the expected future behavior. Formally, this leads to the problem of estimating a nested expectation: $\mathbb{E}[g(X, \mathbb{E}[\phi(X, Y) | X])]$, where X may represent the price tomorrow, and Y represents the price in a week. The function g captures the dependence on both the current situation and the anticipated future. Besides option pricing, this quantity arises in many other decision-making scenarios. Concrete examples include Bayesian experimental design (Goda et al. 2022), and conditional stochastic optimization (Hu et al. 2024). A variant of Multilevel Monte Carlo (Blanchet and Glynn 2015; Giles and Goda 2019) achieves the optimal classical complexity of $O(1/\varepsilon^2)$ under mild assumptions. In contrast, the quantum-inside-quantum algorithm introduced by Blanchet et al. (2025) achieves the quantum-optimal complexity of $\tilde{O}(1/\varepsilon)$.

5 CASE STUDY: CREDIT RISK ANALYSIS VIA QISKIT

Now we illustrate how to exploit quantum approaches to Monte Carlo to analyze credit risk using the QISKIT package. The problem considered here is still simpler than real-world problems. Nevertheless, we will provide a representative examination of how these circuits are constructed and implemented, and how amplitude estimation is performed using current software. We follow the setting of Egger et al. (2020) and the online tutorial (Qiskit-Ecosystem 2024), while also providing additional explanations.

5.1 Problem Setup

We start by considering the *Gaussian conditional independence (GCI) model* of Rutkowski and Tarca (2015) which defines a joint distribution over $(Z, X_1, \dots, X_K) \in \mathbb{R} \times \{0, 1\}^K$ as follows. Let Z be a standard Gaussian random variable, and let X_1, X_2, \dots, X_K be binary-valued random variables that are conditionally independent given Z . The conditional distribution is then defined by $\mathbb{P}(X_k = 1 | Z = z) := F\left(\frac{F^{-1}(p_k^0) - \sqrt{\rho_k} z}{\sqrt{1 - \rho_k}}\right)$, where $p_k^0 \in [0, 1]$ and $\rho_k \in (0, 1)$ are given parameters, and F denotes the cumulative distribution function of the standard normal distribution. We denote $\mathbb{P}(X_k = 1 | Z = z)$ by $p_k(z)$, $k = 1, \dots, K$.

In financial applications, the random variable X_k represents the event that obligor k defaults within the risk measurement horizon. See also Rutkowski and Tarca (2015) for more intuition on the above model. For each k , let λ_k denote the loss given default by obligor k . The total loss is therefore expressed as $L = \sum_{k=1}^K \lambda_k X_k$. We are interested in estimating the expected loss $\mathbb{E}[L]$. Other quantities of interest include the Value at Risk $\text{VaR}_\alpha := \inf\{x | \mathbb{P}(L \geq x) \leq \alpha\}$, and the Conditional Value at Risk $\text{CVaR}_\alpha := \mathbb{E}[L | L \geq \text{VaR}_\alpha(L)]$. However, we do not discuss their estimation here.

5.2 GCI Circuit Setup

We explain encoding the independent Gaussian model. We use n_z qubits to encode a discretized Gaussian variable Z , and allocate one qubit for each X_i . Thus the circuit will output the following quantum state: $|\Phi\rangle = \sum_{i=1}^{2^{n_z}} \sqrt{p(z_i)} |z_i\rangle \otimes_{k=1}^K \sqrt{p_k(z_i)} |1\rangle + \sqrt{1-p_k(z_i)} |0\rangle$, where the discrete variable takes on values z_1, z_2, \dots with probability $p(z_1), p(z_2), \dots$, respectively.

We set the upper and lower bounds of the discretized Gaussian, and the parameters p_k^0 and ρ_k for the GCI model. The circuit can be constructed directly using the built-in class provided by Qiskit. For example, we set $n_z = 4$, and the truncation threshold between -3 to 3 for the discretized Gaussian. We also follow Egger et al. (2020) and set $K = 2, p_1^0 = 0.15, p_2^0 = 0.25, \rho_1 = 0.1, \rho_2 = 0.05, \lambda_1 = 1, \lambda_2 = 2, \alpha = 0.05$ for the GCI model. The circuit can then be constructed as follows:

```
import numpy as np
from qiskit.circuit.library import GaussianConditionalIndependenceModel
n_z = 4
z_max = 3
z_values = np.linspace(-z_max, z_max, 2**n_z)
p_zeros = [0.15, 0.25]
rhos = [0.1, 0.05]
lgd = [1, 2]
alpha = 0.05

#construct the circuit
GCI = GaussianConditionalIndependenceModel(n_z, z_max, p_zeros, rhos)
```

Running this circuit followed by measuring all the qubits yields a classical bitstring of length $n_z + K$, where the first n_z bits represent the discretized Gaussian Z , and the remaining K bits correspond to the realization of (X_1, \dots, X_K) . The constructed circuit can be executed using the `Sampler` class in `Qiskit`.

```
GCI_measure = GCI.measure_all(inplace=False)
sampler = Sampler()
## executing the GCI circuit independently 2,000 times
job_1 = sampler.run(GCI_measure, shots = 2e3)
job_1.result()
```

The above code runs the GCI circuit independently 2,000 times, generating 2,000 i.i.d. binary vectors. As a sanity check, we set `shots = 2,000` and `20,000`, respectively, and plot the histogram of the discretized Z in Figure 3 (Left). As observed, the discretized Z takes values in the range $(-z_{\max}, (-1 + 2^{-n_z+1})z_{\max}, \dots, z_{\max})$. As the number of samples increases, the histogram gradually approaches the shape of a bell curve.

5.3 Amplitude Estimation in Action

To apply quantum versions of Monte Carlo for estimating statistics associated with the total loss L , the next step is to design two circuits: one that encodes the loss function $L: \{0, 1\}^K \rightarrow [0, \infty)$; and another that maps the loss value to the amplitude of an auxiliary qubit. The “quantum” way to encode such a function is using the following circuit: $\mathcal{S}: |x\rangle |0\rangle_{n_S} \rightarrow |x\rangle |L(x)\rangle_{n_S}$.

Here, $|x\rangle = |x_1, \dots, x_K\rangle$ represents a register of K qubits, analogous to the classical n -bit input of the function L , and $|\cdot\rangle_{n_S}$ denotes a register of n_S qubits. If $\lambda_1, \dots, \lambda_K$ are all positive integers, then one can take $n_S = \lceil \log_2(\lambda_1 + \lambda_2 + \dots + \lambda_K) \rceil + 1$ since all possible outcomes of L can be expressed in binary expansion.

Next, we need a circuit that maps L to the amplitude of an auxiliary qubit: $\mathcal{C}: |L\rangle_{n_S} |0\rangle \rightarrow |L\rangle_{n_S} \left(\sqrt{\frac{L_{n_S}}{2^{n_S}-1}} |1\rangle + \sqrt{1 - \frac{L_{n_S}}{2^{n_S}-1}} |0\rangle \right)$. Therefore, estimating $\mathbb{E}[L]$ is equivalent to estimating the amplitude

of the state $|1\rangle$ in the final qubit

$$\left(\sqrt{\frac{L_{n_s}}{2^{n_s} - 1}} |1\rangle + \sqrt{1 - \frac{L_{n_s}}{2^{n_s} - 1}} |0\rangle \right), \quad (1)$$

and then multiply by the normalizing factor $2^{n_s} - 1$. Both circuits are available in `qiskit.circuit.library`.

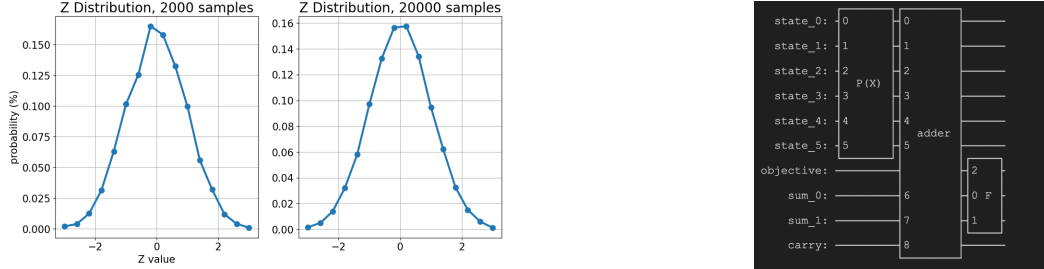


Figure 3: Left: Histogram of Gaussian variable Z. Right: Our quantum circuit.

Our full circuit has three components: we first apply the GCI circuit to encode the underlying distribution, then use the adder circuit \mathcal{S} to encode the loss L , and finally apply the amplitude circuit \mathcal{C} to encode L into the amplitude. The overall circuit is their composition, $\mathcal{C} \circ \mathcal{S} \circ \text{GCI}$. We can also draw the whole circuits in qiskit; see Figure 3 (Right). The leftmost $P(X)$ circuit encodes the GCI model. The middle adder circuit maps the inputs (X_1, X_2) to the loss value $L(X_1, X_2)$. The final circuit 'F', encodes the loss L into the amplitude of the objective qubit (1), which is the 7th qubit from top to bottom. Finally, we apply the QAE algorithm to estimate the expectation and report both the point estimate and the corresponding confidence interval.

```
from qiskit_algorithms import IterativeAmplitudeEstimation, EstimationProblem
ae = IterativeAmplitudeEstimation(
    epsilon_target=epsilon, alpha=alpha,
    sampler=RefSampler(options={"shots": 1000, "seed": 1}))
result = ae.estimate(problem)
conf_int = np.array(result.confidence_interval_processed)
print("Estimated_value:\t%.4f" % result.estimate_processed)
print("Confidence_interval:\t\t[%.4f, %.4f]" % tuple(conf_int))
```

In this example, the estimated value is 0.6681, with a 95% confidence interval of $[0.6435, 0.6928]$. It aligns well with the ground truth value of 0.6446.

6 CONCLUSION AND FURTHER RESOURCES

In this tutorial, we have set the stage by explaining why quantum computing is attracting interest and how its different architectures (gate-model vs. annealing) operate and require error correction. We covered the basic elements of quantum computing using Grover's algorithm to illustrate superposition, interference, and the structure of quantum algorithms. Building on that, we explored quantum approaches to Monte Carlo methods and showed how a variation of Grover's search (QAE) can estimate expected values with a quadratic speedup. A financial example demonstrated how QAE might be applied to nested expectation problems in risk analysis or option pricing, with a sketch of a Qiskit implementation provided for clarity.

We believe that research on quantum speedup of Monte Carlo methods is worthwhile pursuing. There are basic features that we take for granted in classical algorithms which are not as easy to translate to quantum algorithms without incurring a significant computational overhead. For instance, acceptance/rejection is a common practical and generic tool for sampling generation. Of course, there are many settings, especially in high dimensions in which acceptance/rejection is not practical and advanced sampling techniques are

needed. In the quantum setting, state preparation is in general not immediate even for moderate dimensions and there is no direct general and practical tool that at least in principle yields a desired target distribution.

ACKNOWLEDGMENTS

Blanchet, Szegedy, and Wang gratefully acknowledge support from grant NSF-CCF-2403007.

REFERENCES

- Aaronson, S., and P. Rall. 2020. “Quantum Approximate Counting, Simplified”. In *Symposium on Simplicity in Algorithms*, edited by M. Farach-Colton and I.L. Gørtz, 24–32. Philadelphia: SIAM.
- Aharonov, D., and M. Ben-Or. 1997. “Fault-Tolerant Quantum Computation with Constant Error”. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, 176–188. New York, NY, USA: ACM.
- Albash, T., and D. A. Lidar. 2018. “Adiabatic Quantum Computation”. *Reviews of Modern Physics* 90(1):015002(1-64).
- An, D., N. Linden, J.-P. Liu, A. Montanaro, C. Shao, and J. Wang. 2021. “Quantum-Accelerated Multilevel Monte Carlo Methods for Stochastic Differential Equations in Mathematical Finance”. *Quantum* 5:481(1-37).
- Barenco, A., C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, *et al.* 1995. “Elementary Gates for Quantum Computation”. *Physical Review A* 52(5):3457–3467.
- Benioff, P. 1980. “The Computer as a Physical System: A Microscopic Quantum Mechanical Hamiltonian Model of Computers as Represented by Turing Machines”. *Journal of Statistical Physics* 22(5):563–591.
- Bennett, C. H., E. Bernstein, G. Brassard, and U. Vazirani. 1997. “Strengths and Weaknesses of Quantum Computing”. *SIAM Journal on Computing* 26(5):1510–1523.
- Blanchet, J., Y. Hamoudi, M. Szegedy, and G. Wang. 2025. “Non-linear Quantum Monte Carlo”. *arXiv preprint arXiv:2502.05094*.
- Blanchet, J., M. Szegedy, and G. Wang. 2024. “Quadratic Speed-up in Infinite Variance Quantum Monte Carlo”. *arXiv preprint arXiv:2401.07497*.
- Blanchet, J. H., and P. W. Glynn. 2015. “Unbiased Monte Carlo for Optimization and Functions of Expectations via Multi-level Randomization”. In *2015 Winter Simulation Conference (WSC)*, 3656–3667 <https://doi.org/10.1109/WSC.2015.7408524>.
- Brassard, G., and P. Høyer. 1997. “An Exact Quantum Polynomial-Time Algorithm for Simon’s Problem”. In *Proceedings of the Fifth Israeli Symposium on Theory of Computing and Systems*, June 17–19, Ramat Gan, Israel, 12–23.
- Brassard, G., P. Høyer, M. Mosca, and A. Tapp. 2002. “Quantum Amplitude Amplification and Estimation”. *Contemporary Mathematics* 305:53–74.
- Bravyi, S., and A. Kitaev. 2005. “Universal Quantum Computation with Ideal Clifford Gates and Noisy Ancillas”. *Physical Review A* 71(2):022316(1-14).
- Calderbank, A. R., E. M. Rains, P. M. Shor, and N. J. Sloane. 1998, July. “Quantum Error Correction via Codes over GF(4)”. *IEEE Trans. Inf. Theor.* 44(4):1369–1387.
- Calderbank, A. R., and P. W. Shor. 1996. “Good Quantum Error-Correcting Codes Exist”. *Physical Review A* 54(2):1098–1105.
- Cornelissen, A., Y. Hamoudi, and S. Jerbi. 2022. “Near-Optimal Quantum Algorithms for Multivariate Mean Estimation”. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, June 20–24, Rome, Italy, 33–43.
- De Wolf, R. 2019. “Quantum Computing: Lecture Notes”. *arXiv preprint arXiv:1907.09415*.
- Egger, D. J., R. G. Gutiérrez, J. C. Mestre, and S. Woerner. 2020. “Credit Risk Analysis using Quantum Computers”. *IEEE Transactions on Computers* 70(12):2136–2145.
- Einstein, A., B. Podolsky, and N. Rosen. 1935. “Can Quantum-Mechanical Description of Physical Reality be Considered Complete?”. *Physical Review* 47:777–780.
- Farhi, E., J. Goldstone, and S. Gutmann. 2014. “A Quantum Approximate Optimization Algorithm”. *arXiv preprint arXiv:1411.4028*.
- Feynman, R. P. 1982. “Simulating Physics with Computers”. *International Journal of Theoretical Physics* 21(6/7):467–488.
- Giles, M. B. 2015. “Multilevel Monte Carlo Methods”. *Acta Numerica* 24:259–328.
- Giles, M. B., and T. Goda. 2019. “Decision-Making under Uncertainty: Using MLMC for Efficient Estimation of EVPP”. *Statistics and Computing* 29:739–751.
- Goda, T., T. Hironaka, W. Kitade, and A. Foster. 2022. “Unbiased MLMC Stochastic Gradient-based Optimization of Bayesian Experimental Designs”. *SIAM Journal on Scientific Computing* 44(1):A286–A311.
- Gottesman, D. 1997. *Stabilizer Codes and Quantum Error Correction*. Ph. D. thesis, California Institute of Technology.
- Grinko, D., J. Gacon, C. Zoufal, and S. Woerner. 2021. “Iterative Quantum Amplitude Estimation”. *npj Quantum Information* 7(1):52(1-6).
- Grover, L. K. 1996. “A Fast Quantum Mechanical Algorithm for Database Search”. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, May 22–24, Philadelphia, Pennsylvania, USA, 212–219.
- Grover, L. K. 1997. “Quantum Mechanics Helps in Searching for a Needle in a Haystack”. *Physical Review Letters* 79(2):325–328.

- Hamoudi, Y. 2021. *Quantum Algorithms for the Monte Carlo Method*. Ph. D. thesis, Ph. D. thesis, Université de Paris.
- Herman, D., C. Googin, X. Liu, Y. Sun, A. Galda, I. Safro, *et al.* 2023. “Quantum Computing for Finance”. *Nature Reviews Physics* 5(8):450–465.
- Hu, Y., J. Wang, Y. Xie, A. Krause, and D. Kuhn. 2024. “Contextual Stochastic Bilevel Optimization”. In *Advances in Neural Information Processing Systems* 36, edited by A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt and S. Levine.
- Kandala, A., K. Temme, A. D. Córcoles, A. Mezzacapo, J. M. Chow, and J. M. Gambetta. 2019. “Error Mitigation Extends the Computational Reach of a Noisy Quantum Processor”. *Nature* 567(7749):491–495.
- Kitaev, A. Y. 1997. “Quantum Computations: Algorithms and Error Correction”. *Russian Mathematical Surveys* 52(6):1191–1249.
- Kitaev, A. Y. 2003. “Fault-Tolerant Quantum Computation by Anyons”. *Annals of Physics* 303(1):2–30.
- Knill, E., R. Laflamme, and W. H. Zurek. 1998. “Resilient Quantum Computation: Error Models and Thresholds”. *Proceedings of the Royal Society A* 454(1969):365–384.
- Kothari, R., and R. O’Donnell. 2023. “Mean Estimation When You Have the Source Code; or, Quantum Monte Carlo Methods”. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*, January 22–25, Florence, Italy, 1186–1215.
- Lipton, R. J., and K. W. Regan. 2021. *Introduction to Quantum Algorithms via Linear Algebra*. Second ed. MIT Press.
- Montanaro, A. 2015. “Quantum Speedup of Monte Carlo Methods”. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 471:20150301(1-20).
- Nielsen, M. A., and I. L. Chuang. 2010. *Quantum Computation and Quantum Information*. Cambridge University Press.
- Orús, R., S. Mugel, and E. Lizaso. 2019. “Quantum Computing for Finance: Overview and Prospects”. *Reviews in Physics* 4:100028(1-12).
- Peruzzo, A., J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, *et al.* 2014. “A Variational Eigenvalue Solver on a Photonic Quantum Processor”. *Nature Communications* 5:4213(1-7).
- Qiskit-Ecosystem 2024. “Credit Risk Analysis”.
- Rebentrost, P., B. Gupta, and T. R. Bromley. 2018. “Quantum Computational Finance: Monte Carlo Pricing of Financial Derivatives”. *Physical Review A* 98(2):022321(1-15).
- Rhee, C.-h., and P. W. Glynn. 2015. “Unbiased Estimation with Square Root Convergence for SDE Models”. *Operations Research* 63(5):1026–1043.
- Rutkowski, M., and S. Tarca. 2015. “Regulatory Capital Modeling for Credit Risk”. *International Journal of Theoretical and Applied Finance* 18(05):1550034(1-35).
- Shor, P. W. 1994. “Algorithms for Quantum Computation: Discrete Logarithms and Factoring”. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, November 20–22, Santa Fe, NM, USA, 124–134.
- Shor, P. W. 1995. “Scheme for Reducing Decoherence in Quantum Computer Memory”. *Physical Review A* 52(4):R2493–R2496.
- Shor, P. W. 1997. “Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer”. *SIAM Journal on Computing* 26(5):1484–1509.
- Steane, A. M. 1996. “Error Correcting Codes in Quantum Theory”. *Physical Review Letters* 77(5):793–797.
- Suzuki, Y., S. Uno, R. Raymond, T. Tanaka, T. Onodera, and N. Yamamoto. 2020. “Amplitude Estimation without Phase Estimation”. *Quantum Information Processing* 19:1–17.
- Woerner, S., and D. J. Egger. 2019. “Quantum Risk Analysis”. *npj Quantum Information* 5:1098–1105.
- Zalka, C. 1999. “Grover’s Quantum Searching Algorithm is Optimal”. *Physical Review A* 60(4):2746–2751.

AUTHOR BIOGRAPHIES

JOSE BLANCHET is a Professor of MS&E at Stanford University. His research interests include applied probability, Monte Carlo methods, stochastic optimization and quantum computing. His email address is jose.blanchet@stanford.edu and his website is <https://joseblanchet.com>.

MARK S. SQUILLANTE is a Distinguished Research Scientist in Mathematics of Computation at IBM Research. His research interests broadly concern mathematical foundations of the analysis, modeling and optimization of the design, control and performance of stochastic systems. His email address is mss@us.ibm.com and his website is <https://research.ibm.com/people/mark-squillante>.

MARIO SZEGEDY is a Distinguished Professor in the Department of Computer Science at Rutgers University, New Brunswick. His research primarily focuses on quantum computing, computational complexity theory, and recently on AI. His e-mail address is szegedy@cs.rutgers.edu and his website is <https://people.cs.rutgers.edu/~szegedy/homepage.html>.

GUANYANG WANG is an Assistant Professor in the Department of Statistics at Rutgers University, New Brunswick. His research primarily focuses on Monte Carlo methods, generative AI, quantum computing, and probability. His e-mail address is gw295@stat.rutgers.edu and his website is <https://guanyangwang.github.io/>.