8

Finance

The financial services industry is among those beginning to explore the potential future benefits of quantum computing. Finance has the distinct feature that more powerful and more accurate simulations can lead to direct competitive advantage, in a way that is harder to identify in other industries. In this application area, researchers strive to find quantum speedups for use cases of interest to financial services. A number of use cases have been proposed as candidates for quantum solutions, such as:

- **Derivative pricing** (such as options [949, 867], and collateralized debt obligations (CDOs) [981]): Derivatives are financial instruments that are built upon an underlying asset (or assets) that can depend on the value of the asset in potentially complicated ways. In the derivative pricing problem, one needs to determine a fair price of the financial instrument, which is the price that would be received by the seller or paid by the purchaser when an asset or liability is transferred between market participants in an orderly transaction. Typically, one needs to compute the expected value of the fair purchase price of underlying assets at some later date when pricing a derivative. A similar and related problem is known as *computing the Greeks* [950]. The Greeks of a financial derivative are quantities that determine the sensitivity of the derivative to various parameters in the problem. For example, the Greeks of an option are given by the derivative of the price of the option with respect to some parameter, for example, $\Delta := \partial V/\partial X$, where V is the price of the option and X is the price of the underlying asset.
- **Credit valuation adjustments (CVAs)** [491]: CVA is the problem of determining the fair price of a derivative, portfolio, or other financial instrument while taking into account the purchaser's (potentially poor) credit rating, and the risk of default. CVA is typically given by the difference between the

default risk-free portfolio and the value of the portfolio taking into account the possibility of default.

- **Value at risk (VaR)** [1049]: Many forms of risk analysis can be considered, with VaR being a common example. VaR measures the total value a financial instrument (such as a portfolio) might lose over a predefined time interval within a fixed confidence interval. For example, the VaR of a portfolio might indicate that, with 95% probability, the portfolio will not lose more than $\$Y$. A similar technique works as well for the related conditional value at risk (CVaR) problem.
- **Portfolio optimization** [865]: The goal of portfolio optimization is to determine the optimal allocation of funds into a universe of investable assets such that the resulting portfolio maximizes returns and minimizes risk, while also respecting other constraints.

While there are are many more use cases and several approaches for generating quantum speedups, broadly speaking, many use cases stem from one of two paths to quantum improvements: quantum enhancements to Monte Carlo methods (for simulating stochastic processes), and constrained optimization. In the first case, the approach generally involves encoding a relevant, problem-specific function into a quantum state, and then using quantum amplitude estimation to sample from the distribution quadratically fewer times than classical Monte Carlo methods [773]. In the second case, a financial use case is reduced to a constrained optimization problem, and a quantum algorithm for optimization is used to solve the problem.

Among the use cases studied in these two areas, option pricing and portfolio optimization often serve as archetypal examples of Monte Carlo and constrained optimization problems, respectively, and their associated quantum algorithms have the most follow-up work. Moreover, these two classes of problems comprise a considerable fraction of the classical compute used in the financial services industry. For these reasons, we will focus on these two use cases in this chapter, though the approaches, caveats, and complexities typically translate to other relevant use cases.

In addition to the use cases described above, other areas of interest to the financial services industry include post-quantum cryptography, as well as quantum-secure networking and quantum key distribution. However, many of these topics or their proposed quantum implementations are outside the scope of this discussion. Quantum machine learning is yet another popular use case within quantum approaches to finance, but oftentimes these results are quantum approaches to standard machine learning problems, which are then applied to a financial application. As such, we will also not study machine

learning in this finance-specific chapter and instead refer interested readers to any of the excellent review articles on quantum finance (e.g., [525, 176]) for more details.

The authors are grateful to Nikitas Stamatopoulos for reviewing this section of the survey, to Patrick Rebentrost for reviewing Section 8.1, and to Ashley Montanaro for reviewing Section 8.2.

8.1 Portfolio optimization

Overview

Given a set of possible assets into which one can invest, the problem of portfolio optimization (PO) involves finding the optimal allocation of funds in these assets so as to maximize returns while minimizing risk. The Markowitz model, as it is commonly called, is widely used in the financial industry, owing to its simplicity and broad applicability. Sophisticated constraints, transaction cost functions, and modifications to the problem can be used to model realistic, modern PO problems. Numerically solving these optimization problems is a routine part of existing workflows in financial services operations. Several quantum approaches to solving the PO problem have been proposed, each with their own advantages and drawbacks.

Actual end-to-end problem(s) solved

Consider a set of n investable assets with a fixed total budget. Define $w_i \in \mathbb{R}$ to be the fraction of the total budget that is invested into asset i. Thus, the n-dimensional vector w defines a portfolio. Let r be a known n-dimensional vector denoting the expected return for each of the available assets, that is, the percentage by which the value of each asset is expected to grow over some defined time period. Let $\Sigma \in \mathbb{R}^{n \times n}$ be the covariance matrix governing the random (and possibly correlated) fluctuations in the asset returns away from their mean r. In practice, the input parameters Σ and r can be inferred from historical stock price data, or through more sophisticated analyses. The covariance matrix can be used to define a portfolio's "risk" $w^\intercal \Sigma w$, which is precisely the variance in the returns it generates, assuming the underlying model is accurate. Denote the all-ones vector by $\mathbf{1}$, and for any pair of vectors u, v let $\langle u, v \rangle$ denote the standard inner product between u and v. The goal of the Markowitz formulation of PO is to find the optimal portfolio (i.e., vector of weights w) that either:

- maximizes the expected return subject to a fixed risk parameter σ_0^2

$$\begin{aligned} &\max_w \langle w, r\rangle \\ \text{s.t.} \quad &w^\intercal \Sigma w = \sigma_0^2 \\ &\langle \mathbf{1}, w\rangle = 1 \end{aligned} \tag{8.1}$$

- minimizes risk subject to a fixed return parameter r_0

$$\begin{aligned} &\min_w w^\intercal \Sigma w \\ \text{s.t.} \quad &\langle w, r\rangle = r_0 \\ &\langle \mathbf{1}, w\rangle = 1 \end{aligned} \tag{8.2}$$

- maximizes return and minimizes risk with a tradeoff determined by a parameter known as the "risk aversion parameter" λ:

$$\begin{aligned} &\max_w \langle w, r\rangle - \lambda w^\intercal \Sigma w \\ \text{s.t.} \quad &\langle \mathbf{1}, w\rangle = 1 \end{aligned} \tag{8.3}$$

  or the alternative for the square root of risk (standard deviation rather than variance)

$$\begin{aligned} &\max_w \langle w, r\rangle - q\sqrt{w^\intercal \Sigma w} \\ \text{s.t.} \quad &\langle \mathbf{1}, w\rangle = 1, \end{aligned} \tag{8.4}$$

  where q plays the same role as λ.

Typically, it is satisfactory to find a vector that optimizes the objective function up to additive error ϵ, for some prespecified value of ϵ.

When solving the above Markowitz model formulations of PO, the absence of inequality constraints leads to simpler optimization problems that can be solved with efficient classical approaches. For example, the optimization problem in Eq. (8.2) is a simple quadratic program without complicated constraints, for which one can derive a closed-form expression for w using Lagrange multipliers [760]. More general PO problems that include practically relevant constraints (such as the simple "long-only" constraint $w_i \geq 0$, which prohibits "short" positions in which w_i can be less than zero) cannot generically be solved analytically, and one needs to employ more sophisticated numerical solvers. Real-world PO problems include a number of possible constraints (see [781] for a discussion), including, but not limited to:

- Long only—$w_j \geq 0$ for all j.
- Investment bands—$w_j \in [w_j^{\min}, w_j^{\max}]$.

- Turnover constraints—$|\Delta w_j| \leq U_j$ for some fixed fraction U_j, where Δw_j represents the change in holdings of asset w_j from one portfolio to the next.
- Cardinality constraints—minimum, maximum, or exact number of nonzero assets in the portfolio.
- Sector constraints—specified minimum and/or maximum allocations to groups of assets (e.g., the energy or healthcare sectors).
- Transaction costs—typically represented as a function of $|\Delta w_j|$, and often added as a term in the objective function rather than as a constraint.
- Market impact—the effect on the price of an asset that a market participant has when buying or selling the asset. Related to liquidity, market impact can be seen as a type of transaction cost that arises when a transaction causes the price of the asset to move.

The PO can also be formulated in an "online" manner, where, for example, asset performance data arrives one day at a time, and one has the opportunity to update the portfolio at the end of each day [686].

As is often the case with optimization problems, the problem formulation *strongly* affects the solution strategy and the problem "hardness." If the PO problem is unconstrained and continuous (i.e., each w_i is a real number), then the problem is relatively easy. If convex inequality constraints, such as the long-only or turnover constraints, are imposed, then the problem is harder but can still be tackled by relatively efficient methods for convex optimization. By contrast, if one discretizes the problem (so that w now represents an integer number of asset shares or lots being traded), or if one applies some of the constraints above (such as integer-valued constraints like cardinality), then the problem becomes nonconvex and considerably harder to solve. In general, with discrete constraints, the problem can be formulated as an instance of an integer program (IP) (if all variables are discrete) or a mixed-integer program (MIP) (if some variables are discrete and others are continuous), which are NP-hard and therefore intractable to solve in polynomial-time (in n) under widely believed assumptions. Alternatively, given the IP formulation of the problem as a starting point, one can encode the integer variables in a binary representation, thereby allowing the problem to be formulated as a quadratic unconstrained binary optimization (QUBO) instance [881]. These formulations allow quantum algorithms for combinatorial optimization to be employed; for example, the MIP formulation can be solved with a branch-and-bound approach [247], and the QUBO formulation can be solved via Grover-type methods, or heuristically through (NISQ-friendly) quantum annealing approaches (e.g., [790]).

Dominant resource cost/complexity

An early approach to solving this optimization problem using a quantum algorithm was presented in [865], in which the Markowitz problem is written as minimizing risk with fixed return (Eq. (8.2)), and without other complicated constraints. This simple optimization problem boils down to an equality constrained convex program; it can be solved by introducing Lagrange multipliers and solving a linear system (represented by a matrix G) involving the input data r and Σ [865]. The approach of [865] is to use a quantum linear system solver (QLSS) and prepare the quantum state $|w\rangle$ whose amplitudes are proportional to the optimal weights w_i. The complexity to do so to error ϵ is $\widetilde{O}(\kappa\zeta \log(1/\epsilon))$, where κ is the condition number of the matrix G being inverted and $\zeta = \|G\|_F/\|G\|$ is the ratio of its Frobenius norm to its spectral norm. The $\widetilde{O}$ suppresses logarithmic factors, including a factor coming from applying unitaries that block-encode the matrix G in polylog(n) depth, essentially equivalent to the assumption that log-depth quantum random access memory (QRAM) is available. It is a priori unclear what the value of κ and ζ would be for actual PO instances and whether they depend on n, but the explicit logarithmic dependence of this complexity on n is appealing. However, a drawback of this approach is that it produces the quantum state $|w\rangle$ rather than an estimate for the optimal portfolio w. Learning the n entries of w to precision ϵ in 2-norm incurs multiplicative overhead of $\widetilde{O}(n/\epsilon)$ using quantum pure state tomography [49] for total time complexity $\widetilde{O}(n\kappa\zeta/\epsilon)$.[1]

When convex linear inequality constraints, such as long-only or turnover constraints, are included, the above approach will not work. However, a more sophisticated method can be applied, which first maps the PO instance to a convex program (specifically, a second-order cone program (SOCP)) and then makes use of interior point methods to solve the program. These interior point methods can be quantized, forming quantum interior point methods (QIPMs) [612, 68]. The QIPM is an iterative method, where each iteration involves solving a linear equation with a QLSS and classically reading out the solution with tomography. Thus, the procedure within each iteration is similar to the procedure above for solving the unconstrained PO problem, but the linear system to be solved is different (and changes with each iteration). A preliminary study of the effectiveness of this approach for PO was given by [611], followed by a

[1] Reference [865] suggests several possible nonstandard problems that can be solved with $|w\rangle$ without actually learning the entries of w, such as sampling values of i with large $|w_i|$, and estimating overlaps $\langle \tilde{w}, w\rangle$ with hypothesized portfolios $\tilde{w}$. In general, inner products $\langle u, w\rangle$ of arbitrary normalized vectors u with w can be learned to precision ϵ using overlap estimation [637] (an application of amplitude estimation), incurring multiplicative overhead of $O(1/\epsilon)$, but no explicit linear-in-n dependence. However, the practical utility of such tasks within the existing workflows of financial institutions is unclear.

more extensive study in [328]. The QIPM produces an ϵ-optimal classical estimate for w, and has time complexity $\widetilde{O}(n^{1.5}\zeta\kappa\xi^{-1}\log(1/\epsilon))$, where κ and ζ are the maximum condition number and Frobenius-to-spectral-norm ratio for the matrices that must be inverted over the course of the algorithm, respectively, and ξ is the precision to which tomography must be performed. Note that in principle ξ can stay constant even as the overall precision estimate $\epsilon \to 0$ [328].

With the addition of discrete constraints, PO is instead formulated as a nonconvex MIP. MIPs are typically solved with a branch-and-bound approach (for a summary in a financial context, see, e.g., [311, Chapter 11]). Key to this approach is the ability to solve convex relaxations of the MIP (where the discrete constraints are dropped) in poly(n) time (perhaps via classical or quantum interior point methods for SOCPs, as above). To impose the discrete constraints, a tree is constructed and explored, where generating the children of a given node in the tree requires solving one of these relaxations. Thus, the number of convex relaxations that must be solved is proportional to the tree size T, which is generally exponentially large in n. Reference [247] (extending prior work of [776]) showed that a quantum algorithm can produce the same output while exploring quadratically fewer nodes, solving roughly $\widetilde{O}(\sqrt{T})$ convex relaxations (but doing so coherently, which could introduce overheads), for a total complexity of $\widetilde{O}(\sqrt{T}) \cdot \text{poly}(n)$. The value of T is instance dependent and requires empirical estimation: a preliminary numerical analysis of the value of T for a certain ensemble of PO instances up to $n = 56$ found that $T \sim 2^{0.14n}$ to $2^{0.20n}$ [247].

The algorithm for online PO of [686], which leverages the multiplicative weights update method, has time complexity scaling as $\widetilde{O}(\sqrt{n})$, a potentially quadratic speedup compared to the analogous classical algorithm. However, the quantum algorithm has worse dependence on the number of time steps.

The assessment of the number of qubits used by these algorithms requires a nuanced discussion of loading classical data. A key feature of all of the approaches above is that they require (repeatedly) accessing the classical data representing the historical stock information (i.e., the returns r and the covariance matrix Σ) in the quantum algorithm. The size of this data is typically $O(n^2)$. Loading can be performed using block-encoding dense matrices of classical data and QRAM, which achieves $O(\log(n))$ depth (time), at the expense of $O(n^2)$ space. Here, several caveats are inherited from the QRAM primitive. Moreover, for practical values of n, this $O(n^2)$ space cost could be prohibitively large, although it is possible this space cost could manifest as a dedicated QRAM hardware element of the device, rather than as part of the main processor. If log-depth QRAM of sufficient size is not desired or not

available, the data could instead be loaded with only $O(\log(n))$ space and in $O(n^2)$ time, but this overhead in time would likely preclude the possibility of quantum speedup at least in the first two cases, where the formulation is convex and classical poly(n)-time algorithms exist.

Existing resource estimates

A detailed, end-to-end resource analysis of the PO problem using QIPMs was performed in [328]. The authors followed the approach of [611] and performed a careful accounting of all quantum resources, including constant prefactors. The authors found that one needs $800n^2$ logical qubits, a T-depth of

$$(2 \times 10^8)\kappa\zeta n^{1.5}\xi^{-2} \log_2(n) \log_2(\epsilon^{-1}) \log_2(\kappa\zeta n^{14/27}\xi^{-1}),$$

and a T-count of

$$(7 \times 10^{11})\kappa\zeta n^{3.5}\xi^{-2} \log_2(n) \log_2(\epsilon^{-1}) \log_2(\kappa\zeta\xi^{-1}),$$

where κ is the maximum condition number encountered in the algorithm, ζ is the maximum Frobenius-to-spectral-norm ratio, and ξ is the minimum tomographic precision required. The ξ^{-2} dependence can asymptotically be improved to ξ^{-1} at the expense of a more sophisticated protocol for tomography [49]. Furthermore, these estimates used explicit bounds on the complexity of the QLSS from [313] which have since been improved upon in [572, 327]. Using the updated bounds from [327] would immediately reduce the T-count and T-depth estimates by at least three orders of magnitude. Note also that this calculation incorporated optimized circuits for block-encoding dense matrices of classical data with $O(\log(n))$ T-depth but $O(n^2)$ T-count [296], leading to the large discrepancy between those two quantities. The authors performed numerical simulations of PO instances to determine the instance-specific quantities. Using numerically determined values for $\kappa\zeta$ and ξ, and using realistic values of $\epsilon = 10^{-7}$ and $n = 100$, these resource counts imply that one would need 8×10^6 logical qubits, 2×10^{24} T-depth, and 8×10^{29} T-count. These logical estimates for the number of non-Clifford gates could in principle be turned into estimates for the number of physical qubits and runtime on actual hardware, using the methods discussed in the section on fault-tolerant quantum computation. However, the authors of [328] did not do so, in part because the logical costs were sufficiently high that the qualitative conclusion about the practicality of the algorithm was already clear.

Caveats

The quantum algorithms for PO discussed above inherit many of the caveats of their underlying primitives, namely, QLSS, tomography, and classical data

loading. One salient caveat is that the QLSS-based approaches depend on a number of instance-specific parameters κ, ζ, ξ, which are difficult to predict without running numerical simulations. The asymptotic speedup is subject to assumptions about the scaling of these parameters. Additionally, for a speedup to be possible, log-depth QRAM must be available on large datasets, which, while theoretically possible, presents practical challenges.

The branch-and-bound approach does *not* require log-depth QRAM to achieve its nearly quadratic speedup since the runtime will be dominated by the exponential tree-size factor (although it would help to have fast QRAM to reduce by poly(n) factors the time needed to solve the convex relaxations at each step). However, a caveat to that approach is that to obtain the quadratic speedup, the convex relaxations of the MIP (which would be SOCPs), would need to be solved coherently. In principle, this is always possible, but it would likely require a substantial amount of coherent classical arithmetic and additional poly(n) overheads in time and space.

Comparable classical complexity and challenging instance sizes

Convex formulations of the PO problem are typically solved classically via mapping to SOCPs. Optimized software packages can solve these SOCPs efficiently, and many are based on interior point methods. These interior point methods have theoretical runtime complexity of roughly $\widetilde{O}(n^{\omega+0.5}\log(1/\epsilon))$, where $\omega \approx 2.373$ is the matrix multiplication exponent, although for practical instance sizes, the effective value of ω is typically closer to 3. Note that the example PO problem with 100 assets solved in [328] and described above can typically be solved within seconds on a laptop using traditional classical methods. Problem sizes found in the financial services industry can include as many as tens of thousands of assets.

In the IP or MIP formulation of PO, classical solutions will have complexity exponential in n. As a point of reference, the numerical experiments reported in [247] classically solved hundreds of PO instances up to size $n = 56$ (and likely could have gone significantly higher).

Speedup

Recall that the QIPMs used to solve the SOCP for constrained PO are virtually identical to their classical counterpart; they differ by their use of a quantum subroutine to solve linear systems. Thus, any speedup obtained by the quantum approach to solving the SOCP will necessarily come from speedups from the QLSS plus tomography approach to solving a linear system. The approach for unconstrained PO was also based on the same primitives. The performance of the quantum method is often compared against classical Gaussian elimination.

However, since the quantum approach necessarily produces an approximate solver (due to tomography), another valid comparison to make is against approximate classical solvers, such as the conjugate gradient method [527] or the randomized Kaczmarz method [959]. In the case of the randomized Kaczmarz method, the classical complexity for solving an $L \times L$ linear system to precision ξ scales as $O(L\kappa^2\zeta^2 \log(\xi^{-1}))$ (where κ is the condition number and ζ the Frobenius-to-spectral-norm ratio) compared to $O(L^3)$ for Gaussian elimination (asymptotically $O(L^\omega)$). Thus, the quantum method provides the greatest speedup when $\kappa\zeta \propto L$ and $\xi = O(1)$, in which case the QIPM for constrained PO runtime scales as $\widetilde{O}(n^{2.5})$, whereas the classical runtime scales as $\widetilde{O}(n^{3.5})$, where n is the number of stocks in the portfolio (see [328, Table XI] for a more complete discussion). For unconstrained PO, which only requires solving one linear system, the comparison would be $\widetilde{O}(n^2)$ vs. $\widetilde{O}(n^3)$. In either case, the speedup is subquadratic. Moreover, the numerical simulations in [328] were not consistent with these optimistic assumptions on $\kappa\zeta$ and ξ, suggesting, rather, that the QIPM would have minimal if any speedup over classical IPMs, albeit based on small instance sizes up to $n = 120$.

The speedup for the branch-and-bound approach to the MIP formulation of PO is quadratic (up to log factors), although, as mentioned, in contrast to the convex formulations, both the quantum and classical algorithms generally have runtime exponential in n.

NISQ implementations

Several alternative approaches to PO using quantum solutions have been proposed.

- Hybrid-HHL [1062]. This work generalizes the algorithm of [865], described above, by employing midcircuit measurements and conditional logic to obtain a NISQ version of the QLSS that readily solves the PO problem.
- Variational approaches based on the quantum approximate optimization algorithm (QAOA) [184, 524, 85] . These approaches typically use the quadratic objective function from Eq. (8.3), but instead consider $w_i \in \{0, 1\}$ as binary variables indicating whether or not an asset is part of the portfolio (a substantial deviation from the normal formulation). Constraints are dealt with by adding penalties to the objective function. Alternatively, constraints can be enforced by choosing clever versions of the ansatz [802] or by making measurements to project into the feasible space [524].
- Quantum annealing approaches: [881, 825, 824, 452, 790]. As in the previous case, these approaches require the problem to be formulated as a binary

optimization problem. However, in this case, they typically start with the IP formulation and encode integers in binary through one of several possible encodings [881] (thus, the number of binary variables will be greater than n). Constraints in the PO problem can also be included in the objective function using a variety of tricks, resulting in the desired QUBO, which can then be solved using a quantum annealer.

Outlook

The QIPM approach (and QLSS-based techniques more generally) for continuous formulations of PO have the potential to offer polynomial (but subquadratic) speedups for the PO problem. However, these speedups are subject to conjectures about the scaling of certain instance-specific parameters and preliminary empirical estimates are not suggestive of a maximal speedup. In any regard, the resource estimates of [328] illustrate that the non-Clifford resources required to implement the QIPM for this use case are prohibitive, even at problem sizes that are trivial to solve with classical computers. An asymptotic quantum advantage for this problem could exist for sufficiently large sets of assets, but without drastic improvements to the quantum algorithm and the underlying primitives (e.g., QRAM, QLSS), it is unlikely this approach will be fruitful. Even if such improvements are made, the algorithm only provides a polynomial speedup that is subquadratic, at best, greatly limiting the upside potential of this approach.

The branch-and-bound approach for discrete formulations has the possibility of a larger quadratic speedup, but, as has been observed (see, e.g., [223, 79]) in the context of Grover-like quadratic speedups in combinatorial optimization, it is unclear whether the quadratic speedup is sufficient to overcome the inherently slower quantum clock speeds and overheads due to fault-tolerant quantum computation for practical instance sizes.

8.2 Monte Carlo methods: Option pricing

Overview

Many financial instruments require an estimate of the average of some function of a stochastic variable within a window of time. To compute this average, one can use Monte Carlo methods to perform many simulations of the stochastic process over the time window, evaluate the function (which can potentially depend on the path taken by the stochastic variable during the entire window), and numerically estimate the average. While the setup and details of the problems may vary from one use case to another, the underlying methods are often

quite similar. As an archetypal example of this problem, we will focus on the problem of pricing derivatives, such as options, but we remark that many of these results can be carried over to other use cases, such as computing the Greeks, credit valuation adjustments, and value at risk.

Derivatives are financial instruments that, roughly speaking, allow the parties involved to benefit when an asset (such as a stock) increases or decreases in value, but without having to hold the asset itself. One type of derivative—called an "option"—is a contract that permits the holder to either purchase ("call option") or sell ("put option") an underlying asset at a fixed, predetermined price (the "strike price") at or prior to some predetermined time in the future ("the exercise window"). The seller of the option is obligated to either sell or buy the asset, should the holder choose to exercise the option.

How, then, should one decide on a price for the option (i.e., the amount the holder must pay for the contract, not the strike price)? The well-known Black–Scholes (or Black–Scholes–Merton) model provides one approach to pricing options, making a few assumptions about the underlying assets and the rules of the contract. More complicated options can be considered that include, for example, multiple assets in the contract (e.g., basket options), multiple possible exercise windows (e.g., Bermudan or American options), etc.

Typically, options are priced by running Monte Carlo sampling on the value of the underlying asset(s) and determining the expected profit or loss from a given position, which can be translated into a price that the purchaser must pay. Options with a larger potential downside for the seller should cost a larger amount to purchase. For more information on options and Monte Carlo methods in the context of computational finance, see [554, 435].

Actual end-to-end problem(s) solved

The task is to price an option based on an underlying asset. The price of the asset is a random variable X that follows a known (or assumed) stochastic process that models the market for the underlying asset. The option has a known payoff function $f(X)$ (e.g., the difference between the price of the asset at each time step minus the strike price over the trajectory, or zero, whichever is larger). For options that depend on more than one underlying asset or on asset prices at multiple distinct points in time, the random variable X would represent a vector of data containing all information needed to compute the payoff. Given these inputs, the end-to-end problem is to compute an estimate of the expected payoff $\mathbb{E}_X(f(X))$ that lies within a certain error tolerance ϵ with high probability. This quantity is then used to determine the fair price of the option, which we take to be the expected value of the derivative at the contract's expiration date, discounted to the pricing date.

Using the assumed stochastic model for the price of the asset, one can develop a stochastic differential equation for the average payoff of the option. In limited cases, one can compute the average payoff analytically, as in the case of the famous Black–Scholes formula for the price of European call options, for which the 1997 Nobel Prize in Economics was awarded. The Black–Scholes differential equation for the price of an asset at time t can be derived by assuming the price of the underlying stock follows a geometric Brownian motion

$$\mathrm{d}X_t = X_t \alpha \mathrm{d}t + X_t \sigma \mathrm{d}W_t \,,$$

where X_t is the price of the underlying asset at time t, α is a parameter known as the "drift" of the asset, σ is the volatility (the standard deviation of the underlying returns), and $\mathrm{d}W_t$ is an increment of an accompanying Brownian motion W_t. Using Itô's lemma, one can derive a differential equation for the price of the option at time t and, in limited cases (with several assumptions), one can solve the differential equation analytically. In practice, however, different types of contract have more complex definitions and fewer assumptions and, as a consequence, the differential equation cannot be solved analytically. Quantum approaches to numerically solving the stochastic differential equation have been proposed, including finite difference methods [767], Hamiltonian simulation [441], and quantum random walks [692]. For more detail on quantum approaches to solving differential equations, see Chapter 7 on solving differential equations. In many real-world derivative pricing use cases, the underlying differential equation becomes intractable. Thus, the most common classical method of computing the average payoff of an option is not through solving the stochastic differential equation, but rather through Monte Carlo sampling the random process X directly. To do so, one generates a large number of price trajectories over the chosen time range, and the average payoff is computed numerically. In what follows, we will focus on quantum approaches to Monte Carlo estimation—also known as quantum-accelerated[2] Monte Carlo methods—which was pioneered in [773] and subsequently applied to several problems in finance (e.g., [867, 1049, 949, 596, 950, 491]). However, we remark that other approaches to solving this problem that do not make use of Monte Carlo methods have also been proposed (e.g., [868]), and that this is an area of active research.

To compute different quantities, such as value at risk or credit valuation adjustments, similar approaches are often employed: simulate the underlying stochastic evolution several times and estimate the desired quantity numeri-

[2] Not to be confused with quantum Monte Carlo methods, which are classical algorithms for simulating certain quantum systems.

cally. The function to be computed may be quite different, but the approach is often the same.

Dominant resource cost/complexity

In [867, 949], the quantum speedup of Monte Carlo estimation from [773] is applied to solve the option pricing problem. We briefly explain the method and its dominant cost. First of all, this requires discretizing the set of values the random variable X can take, which we index by the label x. Let N denote the number of values and $n = \lceil \log_2(N) \rceil$ denote the number of qubits needed to hold the state $|x\rangle$. The first step is to load the probabilities for the future prices of the asset into the amplitudes of a quantum state, that is, the state

$$\sum_x \sqrt{p_x}|x\rangle\,,$$

where p_x is the probability that x is observed in the corresponding classical Monte Carlo simulation.

Second, a subroutine is employed that computes information about the payoff function into an ancilla register using coherent arithmetic. More precisely, the angle θ_x is computed (rounded to some finite number of bits of precision), where $\sin(\theta_x) = \sqrt{f(x)}$. (For simplicity, here we assume $0 \le f(x) \le 1$ for all x, but we revisit this point later.) This yields

$$\sum_x \sqrt{p_x}|x\rangle|\theta_x\rangle\,.$$

Third, the amplitude $\sqrt{f(x)}$ is loaded into the amplitude of an ancilla register by applying the map $|\theta\rangle|0\rangle \mapsto |\theta\rangle(\sin(\theta)|0\rangle + \cos(\theta)|1\rangle)$. This gives

$$\left(\sum_x \sqrt{p_x f(x)}|x\rangle|\theta_x\rangle\right)|0\rangle + \left(\sum_x \sqrt{p_x(1-f(x))}|x\rangle|\theta_x\rangle\right)|1\rangle\,.$$

The probability of measuring the final ancilla in $|0\rangle$ is precisely $\mathbb{E}_X(f(X))$. Thus, the final step is to apply quantum amplitude estimation, which requires $O(\epsilon^{-1})$ calls to the unitary that produces the state above to obtain an estimate to error ϵ.

If $0 \le f(x) \le 1$ does not hold, the above approach needs to be modified, for example, by shifting and rescaling f over a sequence of intervals of increasing length, as discussed in [773, 867]. Roughly speaking, to make sure that $f(x)$ falls within the interval $[0, 1]$, at least for a large fraction of the randomly chosen values of x, we should expect the function f will need to be scaled down by a factor on the order of the standard deviation $\sigma = \sqrt{\mathbb{E}_X(f(X)^2) - (\mathbb{E}_X f(x))^2}$. Thus, to achieve error ϵ, quantum amplitude estimation must be performed to precision ϵ/σ instead of ϵ.

There are three components to the algorithm that each contribute to the resource cost:

- Loading the distribution with amplitudes $\sqrt{p_x}$. The gate complexity of this step is roughly the same as the time complexity of classically drawing a Monte Carlo sample, although for certain distributions it could be faster (e.g., a quadratic quantum speedup can be obtained if p_x is the stationary distribution of a Markov process [974]). Alternatively, if a functional form for p_x is known, the methods of [754] could be used to approximately prepare the state—note that the Grover–Rudolph approach to state preparation [463] is incompatible with a quantum speedup in the context of Monte Carlo estimation [521]. Finally, [949] proposes using a quantum generative adversarial network (qGAN), a variational quantum algorithm, which could reduce the resources but requires a training phase.
- Coherent arithmetic to compute the rotation angle θ_x. This depends on the complexity of the function f, but can generally be accomplished in comparable gate complexity as classical arithmetic, that is, poly(n). In [948], it was shown how the payoff can instead be put directly into the amplitude, without ever computing θ_x, using quantum signal processing methods [754].
- Quantum amplitude estimation to precision ϵ/σ, which requires $O(\sigma/\epsilon)$ repetitions of the above two costs to achieve an ϵ-estimate on the quantity $\mathbb{E}_X f(X)$.

Overall, from [642, Theorem 1.1] the complexity is

$$\frac{\sigma}{\epsilon} \cdot \text{poly}(n)\,, \tag{8.5}$$

with the poly(n) factor generally on the same order as the time required to draw and process a single classical Monte Carlo sample. This complexity does not require one to have an upper bound on σ, and it improves over the original work of [773] (which in turn is based on the algorithm of [515] for the uniform distribution) and follow-up work [490] by removing additional $\log(\sigma/\varepsilon)$ factors originating from the need to rescale the (potentially unbounded) random variable, to account for the contribution of its tails. In fact, the method can also work even for random variables with infinite variance [156]. However, in practice, the more advanced techniques and analyses of [642, 156] may not be necessary, as the underlying assets are typically modeled with distributions such as Gaussians where the tails are well behaved and the variance of the relevant random variable is controlled.

The general approach to Monte Carlo estimation sketched above has been extended and optimized in various ways; see, for example, [768, 309, 522,

361, 14]. For instance, in [361], the authors study a quantum algorithm for the optimal stopping problem by developing a quantum version of least-squares Monte Carlo. The algorithm finds a quadratic speedup over related classical methods, thereby demonstrating that American-style options—which are more complex than European-style options because they allow the holder to exercise the option and buy/sell the underlying asset at any point in the exercise window, rather than just the end—also maintain a quadratic speedup over classical Monte Carlo.

Existing resource estimates

Detailed resource estimations for benchmark option pricing problems (known as autocallable and target accrual redemption forward, or TARF) were studied in [246]. The authors studied real-world use cases and problem sizes that are simple enough to analyze, yet complex enough to capture desirable features (such as path dependence and multiple underlyings), making them relevant to current financial institutions. For a basket autocallable with 3 underlying assets, 5 payment days, and a knock-in put option with 20 barrier dates, the authors found that one would need about 8000 logical qubits, a T-depth of 5.4×10^7, and a T-count of about 1.2×10^{10}, using the most efficient methods they studied. For a TARF with 1 underlying and 26 payment dates, one needs about 1.2×10^4 logical qubits, a T-depth of about 8.2×10^7, and a T-count of about 9.8×10^9. A follow-up analysis [948] involving a quantum signal processing approach subsequently reduced these estimates to 4.7×10^3 logical qubits, 4.5×10^7 T-depth, and 2.4×10^9 T-count. For comparison, classical Monte Carlo methods are roughly estimated to require 1–10 seconds for 4×10^4 samples to achieve the same accuracy on these examples.

Similar analyses were performed in [950] for the computation of the Greeks, which are quantities that measure the sensitivity of a derivative to various parameters. To compute the Greeks of an option, one needs to compute the derivative of the payoff function with respect to, for example, the price of the underlying. To to do this on a quantum computer, one needs to be able to estimate both the expectation of the payoff function and have a way of computing gradients. The authors apply several quantum methods of computing gradients in order to calculate the Greeks, in addition to the quantum approaches to Monte Carlo methods used. Using a quantum gradient method to compute Greeks of an option, the authors estimate that one would need about 1.2×10^4 logical qubits and a T-depth of around 10^8.

Caveats

There are many types of options and derivatives that may not be accurately captured by these simple models. Some payoff functions are path dependent, and hence one cannot simply use the asset value at some fixed time to compute the cost, but rather the cost depends on the trajectory the random variable takes in each Monte Carlo sample.

Moreover, classical approaches to Monte Carlo sampling often allow for massive parallelization, as each simulation of the underlying asset can be done independently. By contrast, quantum algorithms for this problem require a *serial* approach, as the subroutines in the quantum algorithm must be run one after another without measurement and restart if the quadratic advantage is to be realized. When the slower clock speed found in quantum devices is also taken into account, the requirements for a quantum speedup over classical methods become more stringent, as much larger problem sizes are required to achieve practical advantage. For further reading, see [176, Section 2.3], for example.

It is worth noting that in certain cases the number of classical samples needed to achieve error ϵ can be reduced from the naive $O(\sigma^2/\epsilon^2)$, cutting into the quadratic quantum speedup. In particular, quasi–Monte Carlo methods, which sample possible trajectories of the underlying assets nonrandomly, can achieve a nearly quadratic speedup compared to traditional classical Monte Carlo methods, but gain an exponential dependence on the number of underlying assets ("curse of dimensionality") which limits their use; see [435, Chapter 5]. The number of samples can also potentially be reduced classically via multilevel Monte Carlo methods [426], although a quantum algorithm for multilevel Monte Carlo also exists [33]. In general, when and how these various methods work is delicate and must be evaluated on a case-by-case basis.

Comparable classical complexity and challenging instance sizes

Classical approaches to option pricing comprise some of the largest computational costs incurred by financial institutions. In the traditional approach to solving the option pricing problem, Monte Carlo sampling is required to simulate the evolution of the underlying asset over the time horizon of the option, and it can be slow to converge. In particular, denote the expectation value of $f(X)$ by $V := \mathbb{E}_X(f(X))$, and the variance of $f(X)$ by σ^2. Classical Monte Carlo methods compute an estimate $\hat{V}$ for V formed by averaging $f(X)$ for M independent samples of X. By Chebyshev's inequality,

$$\Pr(|V - \hat{V}| \geq \epsilon) \leq \frac{\sigma^2}{M\epsilon^2}.$$

Thus, classically one needs $M \sim O(\sigma^2/\epsilon^2)$ samples to find an estimate $\hat{V}$ within a 99% confidence interval [773].

In typical industrial scenarios, options can be priced to sufficient operational precision after roughly a few seconds of runtime, sampling as many as tens of thousands of Monte Carlo trajectories.

Alternatively, a tensor network–based classical approach to option pricing was proposed by [602] that could lead to significant advantages over traditional classical methods in some cases.

Speedup

The classical algorithm requires $M = O(\sigma^2/\epsilon^2)$ samples whereas the quantum algorithm requires only $\widetilde{O}(\sqrt{M}) = \widetilde{O}(\sigma/\epsilon)$ samples. The gate cost of a sample is roughly the same classically and quantumly, and thus the speedup is (nearly) quadratic, inherited from the quadratic speedup of quantum amplitude estimation.

Outlook

In [246], the authors place an upper bound on the resources required for pricing options on quantum computers, and they provide a goalpost for quantum hardware development to be able to outperform classical Monte Carlo methods. In particular, the authors estimate that a quantum device would need to be able to execute about 10^7 layers of T gates per second. Moreover, the code distance for fault-tolerant implementation would need to be chosen large enough to support 10^{10} total error-free logical operations. These requirements translate to a logical clock rate of about 50 MHz that would be needed in order to compete with current classical Monte Carlo methods. This clock speed is orders of magnitude faster than what is foreseeably possible given the current status of physical hardware and currently known methods for performing logical gates in the surface code.

While the resource requirements for pricing of derivatives are quite stringent, this is nevertheless an area of active research. For example, a new "analog" quantum representation of stochastic processes was developed in [177] that can compute ϵ-accurate estimates of time averages (over T time steps) of certain functions of stochastic processes in time $\text{polylog}(T) \cdot \epsilon^{-c}$, where $3/2 < c < 2$, an exponential speedup over classical methods in the parameter T. The analog nature of their method leads to additional caveats, and finding concrete applications of this method remains an interesting open question.