Towards Quantum Advantage in Financial Market Risk using Quantum Gradient Algorithms

Nikitas Stamatopoulos[1], Guglielmo Mazzola[2], Stefan Woerner[2], and William J. Zeng[1]

[1]Goldman, Sachs & Co., New York, NY

[2]IBM Quantum, IBM Research – Zurich

We introduce a quantum algorithm to compute the market risk of financial derivatives. Previous work has shown that quantum amplitude estimation can accelerate derivative pricing quadratically in the target error and we extend this to a quadratic error scaling advantage in market risk computation. We show that employing quantum gradient estimation algorithms can deliver a further quadratic advantage in the number of the associated market sensitivities, usually called *greeks*. By numerically simulating the quantum gradient estimation algorithms on financial derivatives of practical interest, we demonstrate that not only can we successfully estimate the greeks in the examples studied, but that the resource requirements can be significantly lower in practice than what is expected by theoretical complexity bounds. This additional advantage in the computation of financial market risk lowers the estimated logical clock rate required for financial quantum advantage from Chakrabarti et al. [Quantum 5, 463 (2021)] by a factor of ~ 7, from 50MHz to 7MHz, even for a modest number of greeks by industry standards (four). Moreover, we show that if we have access to enough resources, the quantum algorithm can be parallelized across up to 60 QPUs, in which case the logical clock rate of each device required to achieve the same overall runtime as the serial execution would be ~ 100kHz. Throughout this work, we summarize and compare several different combinations of quantum and classical approaches that could be used for computing the market risk of financial derivatives.

1   Introduction

Recently, quantum algorithms have been proposed to accelerate the pricing and risk analysis of financial derivatives [1–5]. These algorithms use quantum amplitude estimation to achieve quadratic advantage compared to the classical Monte Carlo methods that are used in practice for most computationally expensive pricing. Let ϵ_p be the error in pricing. The quantum advantage stems from the runtime of a classical Monte Carlo simulation scaling as $\mathcal{O}\left(1/\epsilon_p^2\right)$, while the quantum algorithms have scaling $\mathcal{O}\left(1/\epsilon_p\right)$ [6].

A related and important financial application is the computation of the sensitivity of derivative prices to model and market parameters. This amounts to computing gradients of the derivative price with respect to input parameters. A primary business use of calculating these gradients is to enable hedging of the market risk that arises from exposure to derivative contracts. Hedging this risk is of critical importance to financial firms [7]. In some cases, gradients can be computed analytically, e.g. when the derivative price has an analytical form. In this work, we consider computing gradients where there is not a closed form for the price and where the classical comparison is to Monte Carlo simulation. Gradients of financial derivatives are typically called *greeks*, as these quantities are commonly labeled using Greek alphabet letters. For k greeks, i.e. for a k-dimensional gradient, *classical finite difference* methods compute the price at multiple points in each parameter dimension for a scaling of $\mathcal{O}\left(k/\epsilon_p^2\right)$.

One approach to quantum acceleration of greek computation is to construct a finite difference approximation of the financial derivative's price on a quantum computer and perform amplitude estimation on that quantity instead of its price. In Sec. 2.4 we show that this approach, called

the *semi-classical* method in [8], scales as $\mathcal{O}\left(k/\epsilon\right)$, where ϵ is the target error for the gradients. However, one can further improve the scaling in the number of greeks k, using quantum algorithms for computing the gradients [8, 9]. A quantum algorithm to compute the gradient's components in superposition was originally introduced by Jordan in Ref. [9] and more recently revisited by Gilyén, Arunachalam, and Wiebe (GAW) [8].

Gilyén et al. [8] perform a rigorous analysis of Jordan's original quantum gradient algorithm [9] to show that in general it scales as $\mathcal{O}\left(\sqrt{k}/\epsilon^2\right)$. They then generalize that algorithm to arbitrary higher order m and show that an order $m = \log(\sqrt{k}/\epsilon)$ quantum gradient algorithm has runtime of $\mathcal{O}\left(\sqrt{k}/\epsilon\right)$ for a specific class of smooth functions. For the sake of simplicity, in the rest of the manuscript we call this result as *GAW quantum gradient* algorithm, from the name of the authors of Ref. [8]. Quantum gradient algorithms have also been previously studied as a subroutine for accelerated convex optimization [10]. Since, in this optimization context, subtleties connected with the error's scaling and parameters setting were found to be important, it is timely to put forward a comprehensive study on the advantage that could be found in financial applications.

In this work, we apply the algorithmic framework from [8] to compute financial greeks and study several varieties of quantum gradient algorithms. We use the quantum pricing method of Chakrabarti et al. [5] as a subroutine to the GAW algorithm to numerically estimate the resource requirements of computing the greeks of two types of option contracts (a) a European call option, which we use as a benchmark to establish the validity of the algorithm and estimate the corresponding resource requirements, and (b) a path-dependent basket option whose pricing profile is representative of typical financial derivative contracts of practical interest.

Then, we introduce a second-order accurate quantum gradient algorithm, the $m = 1$ version of [8], for which we can give an explicit and compact quantum implementation in the financial derivative case, and which does not rely on block encoding or Hamiltonian simulation. We use this *Simulation-Free Quantum Gradient* (SFQG) method to compute the greeks of a path-dependent derivative and find that it is significantly cheaper to construct than the Hamiltonian-based method. Additionally, we show that we can improve the overall performance of quantum gradient estimation algorithms by employing a maximum likelihood (MLE) method to extract the most likely estimate of the gradients with concrete confidence intervals. With these tools, we calculate that quantum advantage for calculating risk may be achievable with quantum computers whose clock rates are 7 times slower than that required for pricing itself. We discuss these implications in more detail in Sec. 7.

Finally, we perform a comparison between quantum, classical and semi-classical gradient estimation algorithms in the context of financial derivatives, summarized in Table 4.

We highlight the new contributions in this work:

- We numerically study the quantum gradient estimation algorithms from [8, 9] for functions of practical interest to financial market risk and compare the observed oracular cost to theoretical expectations. (Section 4)

- We devise a method to construct a second-order accurate oracle for quantum gradient estimation for functions computed using quantum amplitude estimation that is cheaper in required resources compared to existing methods. (Section 5)

- We introduce a way to improve gradient estimation algorithms using classical maximum likelihood estimation (MLE). (Section 6)

- We propose a technique to employ automatic differentiation (AD) methods on quantum computers which can enhance the quantum gradient estimation performance in certain cases. (Appendix B)

- We update the resource estimates for quantum advantage in financial derivative pricing from prior research. (Section 7)

1.1   Quantum Amplitude Estimation

Our gradient calculation algorithms extend the quantum accelerated method for derivative pricing that is based on quantum amplitude estimation (QAE) [11]. We review QAE here. Let the k-

Classical Finite Difference	CFD with CRN	Semi-classical	Jordan's	GAW Quantum Gradient
$\mathcal{O}\left(k/\epsilon^3\right)$	$\mathcal{O}\left(k/\epsilon^2\right)$	$\mathcal{O}\left(k/\epsilon\right)$	$\mathcal{O}\left(\sqrt{k}/\epsilon^2\right)$	$\mathcal{O}\left(\sqrt{k}/\epsilon\right)$

Table 1: Summary of the complexity scaling of the different market risk algorithms. Here ϵ is the absolute error in gradient estimation, and k is the dimension of the gradient. For the CFD and semi-classical methods, which depend additionally on the discretization step h, we assume the optimal choices, which are derived in Section 2.2.

dimensional vector $\mathbf{x}$ denote the set of market data parameters of the derivative represented as a basis state $|\mathbf{x}\rangle$ and $f(\mathbf{x})$ its price, rescaled to satisfy $f(\mathbf{x}) \in [0, 1]$. We assume the existence of a unitary operator $\mathcal{A}$ which produces the state

$$\mathcal{A} : |\vec{0}\rangle |\mathbf{x}\rangle \rightarrow \left(\sqrt{1 - f(\mathbf{x})} |\psi_0(\mathbf{x})\rangle |0\rangle + \sqrt{f(\mathbf{x})} |\psi_1(\mathbf{x})\rangle |1\rangle\right) |\mathbf{x}\rangle, \tag{1}$$

where $|\psi_0(\mathbf{x})\rangle$ and $|\psi_1(\mathbf{x})\rangle$ are arbitrary, normalized quantum states. The state $|\mathbf{x}\rangle$ representing the market data parameters acts only as input to $\mathcal{A}$ and we can ignore it for the remainder of this section.

Explicit constructions of $\mathcal{A}$ with resource estimates for specific path dependent derivative instances are given in Chakrabarti et al. [5]. QAE estimates $f(\mathbf{x})$ with repeated applications of the operator $\mathcal{Q} = \mathcal{A}S_0\mathcal{A}^\dagger S_{\psi_0}$, where $S_0 = \mathbb{I} - 2|\vec{0}\rangle\langle\vec{0}|$ and $S_{\psi_0} = \mathbb{I} - 2|\psi_0(\mathbf{x})\rangle|0\rangle\langle 0|\langle\psi_0(\mathbf{x})|$. The QAE algorithm relies on the fact that, by the construction of $\mathcal{Q}$, the state produced after the application of $\mathcal{A}$ in Eq. (1) can be written in the eigenbasis of $\mathcal{Q}$ [11]

$$\frac{-i}{\sqrt{2}}\left(e^{i\theta(\mathbf{x})}|\psi_+(\mathbf{x})\rangle - e^{-i\theta(\mathbf{x})}|\psi_-(\mathbf{x})\rangle\right) \equiv |\Psi(\mathbf{x})\rangle, \tag{2}$$

where $|\psi_+(\mathbf{x})\rangle$ and $|\psi_-(\mathbf{x})\rangle$ are the eigenvectors of $\mathcal{Q}$ with eigenvalues $e^{\pm 2i\theta(\mathbf{x})}$, and $f(\mathbf{x}) = \sin^2(\theta(\mathbf{x}))$. Let integer $m > 0$ denote the bits of precision with which we want to estimate the amplitude. Applying $H^{\otimes m}$ on an m-qubit register initialized at $|0\rangle_m$ and using the register to control different powers of $\mathcal{Q}$, a phase kickback of the eigenvalues of $\mathcal{Q}$ is induced onto the control register

$$\frac{1}{\sqrt{M}}\sum_{j=0}^{M-1}|j\rangle\mathcal{Q}^j|\Psi(\mathbf{x})\rangle = \frac{1}{\sqrt{M}}\sum_{j=0}^{M-1}e^{2ij\theta(\mathbf{x})}|j\rangle|\Psi_+(\mathbf{x})\rangle - \frac{1}{\sqrt{M}}\sum_{j=0}^{M-1}e^{-2ij\theta(\mathbf{x})}|j\rangle|\Psi_-(\mathbf{x})\rangle, \tag{3}$$

where $|\Psi_\pm(\mathbf{x})\rangle = -ie^{\pm i\theta(\mathbf{x})}|\psi_\pm(\mathbf{x})\rangle/\sqrt{2}$ and $M = 2^m$. Finally, applying an inverse Quantum Fourier Transform on the first register and measuring in the computational basis gives an m-bit approximation of either $\tilde{\theta}_+(\mathbf{x}) = M\theta(\mathbf{x})/\pi$ or $\tilde{\theta}_-(\mathbf{x}) = -M\theta(\mathbf{x})/\pi$. This can then be classically mapped to an estimate for $f(\mathbf{x})$ using

$$\tilde{f}(\mathbf{x}) = \sin^2\left(\frac{\pi\tilde{\theta}_\pm(\mathbf{x})}{M}\right). \tag{4}$$

In order to approximate $\tilde{f}(\mathbf{x})$ within additive error ϵ_p with high probability, QAE requires $M = \mathcal{O}\left(1/\epsilon_p\right)$ invocations to $\mathcal{Q}$. We measure the query complexity of the QAE algorithm by counting the number of calls to the oracles $\mathcal{A}$ and $\mathcal{A}^\dagger$ and this scales as $\mathcal{O}\left(2/\epsilon_p\right)$. In the case of financial derivatives, the function f is the expectation value of the payoff of a derivative contract. We explore this context in the following section.

2   Gradient Methods for Financial Derivatives

In this section we describe several methods for computing the gradients of financial derivatives. The complexities of these approaches are summarized in Table 1.

2.1 Greeks

Let $\mathbf{S}^t \in \mathbb{R}_+^d$ be a vector of values for d underlying assets at time t. Let $(\mathbf{S}^1, ..., \mathbf{S}^T) = \bar{\omega} \in \bar{\Omega}$ be a path of a discrete time multivariate stochastic process describing the values of those assets in time when the current (today's) market values of these assets are $\mathbf{S}^0$. We use both notations for a path in the text. The corresponding probability density function is denoted by $\bar{p}(\mathbf{S}^0, \bar{\omega})$. Let $g(\mathbf{S}^0, \bar{\omega}) = g\left(\mathbf{S}^0, ..., \mathbf{S}^T\right) \in \mathbb{R}$ be the discounted payoff of some derivative on those assets. To price the derivative we calculate

$$\mathbb{E}(g) = \int_{\bar{\omega} \in \bar{\Omega}} \bar{p}(\mathbf{S}^0, \bar{\omega}) g(\mathbf{S}^0, \bar{\omega}) d\bar{\omega} \approx \sum_{\omega \in \Omega} p(\mathbf{S}^0, \omega) g(\mathbf{S}^0, \omega), \tag{5}$$

where we have removed the bar notation to indicate that we have switched from a continuous to a discrete model of prices. Gradients of this price are known as greeks.

Example 1 (Delta) *Consider a single underlying ($d = 1$). We define the gradient of the underlying with respect to the spot price S^0 as*

$$\Delta = \frac{\partial \mathbb{E}(g)}{\partial S^0}. \tag{6}$$

Other commonly used greeks are gradients of the price with respect to the time T (*theta*), the volatility of the underlying model for S (*vega*), the correlation between assets, or other parameters. In general one calculates $\mathbb{E}(g, \mathbf{x})$ for some model and/or market parameters $\mathbf{x}$ and then wishes to compute the set $\{\frac{\partial \mathbb{E}(g)}{\partial \theta_{i=1...k}}\}$. Remarkably, k can be on the order of hundreds or thousands in practical cases. For this reason, a k scaling improvement represents an important advantage in this context.

2.2 Classical Finite Difference

Let $f(\mathbf{x}) = \mathbb{E}(g, \mathbf{x})$ be the pricing function for a fixed payoff g. Classically, we can compute the gradients of a function $f : \mathbb{R}^k \to \mathbb{R}$ using finite-difference methods by sampling the function f over a sufficiently small region h so that expanding f to first order gives a good approximation to $f(\mathbf{x}) \approx f(\mathbf{a}) + (\mathbf{x} - \mathbf{a}) \cdot \nabla f$. The simplest *forward* finite-difference classical scheme to compute the gradients at a point $\mathbf{x}_0$ requires $k + 1$ evaluations of f, one at $\mathbf{x}_0$ and k evaluations displaced from $\mathbf{x}_0$ by h in each dimension. The gradient is then approximated using $\partial f / \partial x_i \approx [f(\mathbf{x}_0 + h\hat{e}_i) - f(\mathbf{x}_0)]/h + \mathcal{O}(h)$, where $\hat{e}_i$ is the ith normalized basis vector. For a more accurate approximation, we can instead use a second-order scheme which requires $2k$ function evaluations, and approximate the gradients with

$$\partial f / \partial x_i \approx \frac{f(\mathbf{x}_0 + \frac{h}{2}\hat{e}_i) - f(\mathbf{x}_0 - \frac{h}{2}\hat{e}_i)}{h} + \mathcal{O}(h^2). \tag{7}$$

Now suppose that the function $f(x)$ is evaluated with a finite accuracy $\delta \geq 0$. The associated error for the forward or central finite differences formulas reads

$$\frac{f(x + h) - f(x - (p-1)h) + \mathcal{O}(\delta)}{ph} + \mathcal{O}(h^p) = \partial_x f(x) + \mathcal{O}(\delta/h + h^p), \tag{8}$$

where $p = 1$ for forward and $p = 2$ for central finite differences. Suppose now that we want to achieve an overall estimation error of $\epsilon > 0$, i.e.

$$\delta/h + h^p \leq \epsilon, \tag{9}$$

where we focus on the asymptotic scaling. While the step size h can be freely chosen, improving the accuracy δ is usually related to increasing computational costs. Thus, we want to maximize δ by setting h while still achieving the target accuracy ϵ. This leads to

$$\delta = (\epsilon - h^p)h = \epsilon h - h^{p+1}. \tag{10}$$

Setting the first derivative of the right-hand-side with respect to h to zero leads to

$$h = (\epsilon/(p+1))^{(1/p)}, \tag{11}$$

which leads to the optimal δ for a target ϵ given by

$$\delta = \mathcal{O}(\epsilon^{1+1/p}). \tag{12}$$

When $f(x)$ is approximated by algorithms relying on sampling, the number of samples for a target approximation error scales as $\delta = \mathcal{O}(1/M^q)$, where M denotes here the number of samples and q depends on the convergence rate of the algorithm i.e., $q = 1/2$ for classical Monte Carlo simulation and $q = 1$ for QAE. Then, combining everything together leads to

$$M = \mathcal{O}(\epsilon^{-(1+1/p)/q}). \tag{13}$$

Thus, using $p = 2$ and $q = 1/2$, we calculate that the complexity of computing k greeks using a central-difference method with Monte Carlo is $\mathcal{O}(k/\epsilon^3)$. We call this approach the *classical finite difference* (CFD) method.

2.3 Finite-Difference with Common Random Numbers

Another approach for computing greeks, when derivative pricing is done classically with Monte Carlo, is to use the second-order central-difference method of Eq. (7), but to perform correlated sampling by using the same random numbers in the Monte Carlo evaluation of both $f(\mathbf{x}_0 + \frac{h}{2}\hat{e}_i)$ and $f(\mathbf{x}_0 - \frac{h}{2}\hat{e}_i)$. This way the statistical fluctuations present in both terms cancel out, effectively removing the overall error dependence on the discretization step h. The complexity of evaluating k greeks in this case is $\mathcal{O}(k/\epsilon^2)$, which is also classically optimal [12]. We call this method the *classical finite difference with common random numbers* (CFD-CRN).

2.4 Semi-classical Quantum Gradients

A straightforward approach to improve the finite-difference method using quantum computation is to use a central-difference formula with quantum amplitude estimation algorithm for each pricing step [1, 4, 5]. This *semi-classical quantum gradient* (SQG) method [8] then scales as $\mathcal{O}(k/\epsilon^{1.5})$, which we get by substituting $p = 2$ and $q = 1$ in Eq. (13). This approach can be improved upon in a similar manner to using CRN for classical finite difference. Instead of computing $\mathbb{E}\left(f(\mathbf{x}_0 + \frac{h}{2}\hat{e}_i)\right)$ and $\mathbb{E}\left(f(\mathbf{x}_0 - \frac{h}{2}\hat{e}_i)\right)$ separately using amplitude estimation, we compute $\mathbb{E}\left((f(\mathbf{x}_0 + \frac{h}{2}\hat{e}_i) - f_\omega(\mathbf{x}_0 - \frac{h}{2}\hat{e}_i))/h\right)$. We can do this by computing the quantity $|(f_\omega(\mathbf{x}_0 + \frac{h}{2}\hat{e}_i) - f_\omega(\mathbf{x}_0 - \frac{h}{2}\hat{e}_i))/h\rangle$ in a quantum register, where f_ω denotes the payoff for each path ω (cf. Sec. 2.1) and then using amplitude estimation on this quantity. In this case the output of amplitude estimation is

$$\sum_{\omega \in \Omega} p(\omega) \left(f_\omega(\mathbf{x}_0 + (h/2)\hat{e}_i) - f_\omega(\mathbf{x}_0 - (h/2)\hat{e}_i)\right)/h \approx \mathbb{E}(\partial_i f(\mathbf{x})), \tag{14}$$

giving us the expectation value of the finite difference approximation of the ith derivative. The advantage of this method is that there are no separate statistical fluctuations associated individually with $f_\omega(\mathbf{x}_0 \pm (h/2)\hat{e}_i)$ and we recover the standard $\mathcal{O}(1/\epsilon)$ scaling of amplitude estimation. For k gradients, we then get an overall complexity of $\mathcal{O}(k/\epsilon)$.

2.5 Quantum Gradients

There are other quantum approaches that use a quantum Fourier transform to compute the gradient with an improved scaling in the dimension k [8, 9]. These algorithms require access to a fractional *phase oracle* for the target function $f : \mathbb{R}^k \mapsto \mathbb{R}$, in our case the pricing function of Sec. 1.1. This oracle, given a point $\mathbf{x} \in \mathbb{R}^k$ and $S > 0$, performs the operation

$$O_{Sf} : |\mathbf{x}\rangle \rightarrow e^{2\pi i S f(\mathbf{x})} |\mathbf{x}\rangle. \tag{15}$$

In order to estimate the k-dimensional gradient of f at point $\mathbf{x}_0$, the oracle is evaluated over a uniform superposition of points $\boldsymbol{\delta}$ in a sufficiently small k-dimensional hypercube $G_{\mathbf{x}_0}^k$ of edge

length l around $\mathbf{x}_0$ where each dimension is discretized using N points with $n = \log N$ qubits, such that

$$\frac{1}{\sqrt{N^k}} \sum_{\boldsymbol{\delta} \in G_{\mathbf{x}_0}^k} O_{Sf} \ket{\boldsymbol{\delta}} = \frac{1}{\sqrt{N^k}} \sum_{\boldsymbol{\delta} \in G_{\mathbf{x}_0}^k} e^{2\pi i S f(\mathbf{x_0} + \boldsymbol{\delta})} \ket{\boldsymbol{\delta}} \approx e^{2\pi i S f(\mathbf{x_0})} \frac{1}{\sqrt{N^k}} \sum_{\boldsymbol{\delta} \in G_{\mathbf{x}_0}^k} e^{2\pi i S \cdot \nabla f_{\mathbf{x}_0} \cdot \boldsymbol{\delta}} \ket{\boldsymbol{\delta}}, \quad (16)$$

assuming $f(\mathbf{x}_0 + \boldsymbol{\delta}) \approx f(\mathbf{x}_0) + \nabla f_{\mathbf{x}_0} \cdot \boldsymbol{\delta}$ for $||\boldsymbol{\delta}|| \ll 1$. As shown in [8, 9], choosing $S = N/l$ and applying the k-dimensional inverse Quantum Fourier Transform on the resulting state, we measure in the computational basis to get an approximation of $\nabla f_{\mathbf{x}_0}$ with precision $\epsilon = \mathcal{O}(1/N)$ and high probability. The complexity of these gradient estimation algorithms depends on the resources required to construct the oracle of Eq. (15) and the size of $G_{\mathbf{x}_0}^k$ required to make the approximation in Eq. (16) sufficiently accurate. As we discuss in the next section in more detail, the computational cost of the gradient estimation method depends on the value of $S = N/l$. Intuitively, the cost of the method increases as the desired precision of the gradient estimate increases (larger N), and the value of l denoting the region we evaluate the function f decreases. While the value of N is an input to the gradient estimation method controlling the desired precision, the required value of l depends on the degree of non-linearity of the function f in the region of evaluation around $\mathbf{x}_0$. The more non-linear the function f is around $\mathbf{x}_0$, the smaller l we need to pick so that the function can be sufficiently well-approximated by the first order Taylor expansion used in Eq. (16).

3 Higher-order methods for quantum gradients

Gilyén et al [8] showed that in order to estimate k gradients of a function f with accuracy ϵ, the oracle in Eq. (15) has to be evaluated $S = N/l \sim D_2 \sqrt{k}/\epsilon^2$ times, where D_2 is an upper bound on the magnitude of the second-order derivatives of f and N, l as defined in Section 2.5. While this gives a quadratically improved scaling in k, it does not improve on the error scaling of the semi-classical method. To improve also the scaling in accuracy ϵ, Ref. [8] introduces higher-degree central-difference schemes.

The $2m$-point central-difference approximation for the gradient of f at $\mathbf{0}$ is given by

$$\nabla f_{(2m)}(\mathbf{0}) = \sum_{\ell=-m}^{m} a_\ell^{(2m)} f(\ell \mathbf{x}), \quad (17)$$

where the coefficients $a_\ell^{(2m)}$ depend on the choice of m and for uniform grid spacing we have $a_\ell^{(2m)} = -a_{-\ell}^{(2m)}$ [13]. A phase oracle for the general $2m$-point scheme of Eq. (17) can be constructed by composing individual fractional phase oracles for each of the $2m$ terms, scaled by the appropriate coefficient $a_\ell^{(2m)}$. This leads to a family of algorithms with different phase oracles at different orders m

$$O_{Sf}^m : \ket{\mathbf{x}} \to e^{2\pi i S \sum_{\ell=-m}^{m} a_\ell^{(2m)} f(\ell \mathbf{x})} \ket{\mathbf{x}}, \quad (18)$$

with $S = N/l$. For example, the phase oracle for the two-point approximation

$$O_f^1 \ket{\mathbf{x}} = e^{2\pi i (f(\mathbf{x}) - f(-\mathbf{x}))/2} \ket{\mathbf{x}}, \quad (19)$$

can be constructed as the product of oracles $O_f^+ \ket{\mathbf{x}} = e^{\pi i f(\mathbf{x})} \ket{\mathbf{x}}$ and $O_f^- \ket{\mathbf{x}} = e^{-\pi i f(-\mathbf{x})} \ket{\mathbf{x}}$.

With these higher-order methods, they show that estimating the gradients of a class of smooth functions[1] to accuracy ϵ using Jordan's algorithm, can instead be done with $S = \mathcal{O}(\sqrt{k}/\epsilon)$ phase oracle applications by picking a large enough value for m in the central-difference approximation used. Therefore, for this family of smooth functions, gradient estimation using high-order central-difference methods scales quadratically better in the desired accuracy ϵ compared to the original Jordan's algorithm. This is described formally in the following Theorem:

---

[1] Also known as Gevrey class $G^{\frac{1}{2}}$ functions [14]

Theorem 1 (5.4 from [8]) *Let* $\mathbf{x} \in \mathbb{R}^k$, $\epsilon < c \in \mathbb{R}_+$ *be fixed constants and suppose* $f : \mathbb{R}^k \mapsto \mathbb{R}$ *is analytic and satisfies the following: for every* $j \in \mathbb{N}$ *and* $\alpha \in [k]^j$

$$|\partial_\alpha f(\mathbf{x})| \leq c^j j^{\frac{j}{2}}. \tag{20}$$

Using the GAW Algorithm and setting $m = \log(c\sqrt{k}/\epsilon)$ *we can compute an* ϵ-*approximate gradient* $\tilde{\nabla} f(\mathbf{x}) \in \mathbb{R}^k$ *such that*

$$\|\nabla f(\mathbf{x}) - \tilde{\nabla} f(\mathbf{x})\|_\infty \leq \epsilon, \tag{21}$$

with probability at least $1 - \delta$, *using* $\widetilde{\mathcal{O}}\left(\frac{c\sqrt{k}}{\epsilon} \log\left(\frac{k}{\delta}\right)\right)$ *queries to a probability or (fractional) phase oracle of* f.

In addition to the desired $\mathcal{O}(\sqrt{k}/\epsilon)$ scaling, the complexity of the GAW algorithm in Theorem 1 includes a factor of $\log(k/\delta)$ stemming from the fact that we need to extract the medians after the application of the quantum gradient estimation algorithm, with probability of $1 - \delta$. While in the following sections we focus on the $\mathcal{O}(\sqrt{k}/\epsilon)$ factor in order to establish the dominant scaling of the complexity with respect to k, in Sec. 6 we show how we can eliminate the $\log(k/\delta)$ factor by using classical maximum likelihood estimation. This approach not only decreases the overall computational complexity of the algorithm, but also provides us with concrete confidence intervals for a given confidence level.

We summarize in Table 1 the scaling in k and ϵ of the algorithms discussed in this Section.

3.1 Creating phase oracles from probability oracles

Because the function whose gradients we would like to compute is only accessible through a *probability oracle* in the form of Eq. (1), we need to create a corresponding *phase oracle* of Eq. (15) in order to use the quantum gradient method described in this section. To do so, we can use the block encoding technique from Ref. [8]:

Definition 1 (4.4 from [8]) *Suppose that* A *is an operator on a Hilbert space* $\mathcal{H}$, *then we say that the unitary* U *acting on* $\mathcal{H}_{aux} \otimes \mathcal{H}$ *is a block-encoding of* A *if*

$$A = (\langle\vec{0}| \otimes \mathbb{I})U(|\vec{0}\rangle \otimes \mathbb{I}).$$

Intuitively, the block-encoding U is a unitary whose top-left block contains A:

$$U = \begin{bmatrix} A & \cdot \\ \cdot & \cdot \end{bmatrix}. \tag{22}$$

For the probability oracle $\mathcal{A}$ of Eq. (1), the authors of Ref. [8] observe that

$$(\langle\vec{0}| \otimes \mathbb{I})(\mathcal{A}^\dagger(Z \otimes \mathbb{I})\mathcal{A})(|\vec{0}\rangle \otimes \mathbb{I}) = \text{diag}(1 - 2f(\mathbf{x})), \tag{23}$$

and from Definition 1, $U \equiv \mathcal{A}^\dagger(Z \otimes \mathbb{I})\mathcal{A}$ is a block-encoding of a diagonal matrix H with diagonal entries $(1 - 2f(\mathbf{x}))$. With access to this block-encoding, the Hamiltonian simulation method from [15, 16] allows us to implement an ϵ_{phase}-approximation of the unitary e^{itH}, through repeated applications of U and $U^\dagger$, which scales as $\mathcal{O}(|t| + \ln(1/\epsilon_{\text{phase}}))$. Therefore, for appropriately chosen values of t, we can use block-encoding and Hamiltonian simulation to produce an ϵ_{phase}-approximation of the phase oracle O_{Sf} from Eq. (15). Note that while the block-encoding of Eq. (23) allows us to create a phase oracle with phase of $1 - 2f(\mathbf{x})$ instead of $f(\mathbf{x})$ in Eq. (15), the first term ends up as a global phase which can be ignored, and the factor of -2 can be absorbed in the factor t of the Hamiltonian simulation.

While references [15, 16] describe how to perform Hamiltonian simulation and the resources required to realize the necessary unitary evolution, the methods presented therein require post-selection, which needs to be factored in for an end-to-end resource estimation. On the other hand, Ref. [17] introduces a coherent Hamiltonian simulation method which does not require post-selection and instead succeeds with arbitrarily high probability $1-\delta$, scaling as $\mathcal{O}(|t|+\ln(1/\epsilon_{\text{phase}})+$

$\ln(1/\delta))$. For a target t, approximation error ϵ_{phase} and $\delta = 2\epsilon_{\text{phase}}$, this coherent Hamiltonian simulation algorithm queries U and its inverse a total number of times

$$N_U(t, \epsilon_{\text{phase}}, \beta) = 2 \left\lfloor \frac{1}{2} r \left(\frac{e|t|}{2\beta}, \frac{5\epsilon_{\text{phase}}}{24} \right) \right\rfloor + \gamma \left(\frac{\epsilon_{\text{phase}}}{3}, 1 - \beta \right) + 1, \tag{24}$$

where

- $\beta \in (0, 1)$ is a user-chosen parameter,

- $r(\tau, \epsilon) = |\tau| e^{W(\ln(1/\epsilon)/|\tau|)}$, where $W(x)$ is the Lambert-W function,

- $\gamma(\epsilon, \Delta) = 2 \cdot \left\lceil \max \left(\frac{e}{\Delta} \sqrt{W \left(\frac{8}{\pi \epsilon^2} \right) W \left(\frac{512}{e^2 \pi \epsilon^2} \right)}, \sqrt{2} W \left(\frac{8\sqrt{2}}{\sqrt{\pi} \Delta \epsilon} \sqrt{W \left(\frac{8}{\pi \epsilon^2} \right)} \right) \right) \right\rceil + 1.$

Using this method, the total number of oracle (Eq. (1)) calls required to construct an ϵ_{phase}-approximation of the m-order phase oracle of Eq. (18) is then given by

$$N_o = \sum_{\ell = -m}^{m} N_U \left(2\pi \frac{N}{l} |a_\ell^{(2m)}|, \epsilon_{\text{phase}}, \beta \right). \tag{25}$$

The optimal value of β depends on the target approximation error ϵ_{phase}. For the cases studied in this manuscript, we find that $\beta = 0.5$ is the optimal choice and we fix that value whenever we use Eq. (25).

4 Resource Estimation of Quantum Gradient Methods

In this section, we perform an asymptotic resource estimation for the gradient methods described previously. We choose representative parameters from the financial domain for gradient estimation problems, targeting $k = 1000$ greeks and an approximation error of $\epsilon = 10^{-3}$. For the resource estimation of the GAW method, we assume that the smoothness conditions of Theorem 1, with smoothness parameter $c = 1$, apply for the problem at hand and that the Hamiltonian simulation phase error is $\epsilon_{\text{phase}} = 10^{-4}$. Here we choose $c = 1$ in order to estimate the possible usefulness of the algorithm in a best-case scenario of smoothness from Theorem 1. While we treat the phase error ϵ_{phase} as a free parameter at this point, our particular choice of 10^{-4} is motivated by numerical simulations which show this to be a good choice. We discuss the numerical simulations and the impact of the phase error in more detail in subsequent sections. Then, we set the finite-difference approximation degree to

$$m = \log(c\sqrt{k}/\epsilon) \tag{26}$$

and the spacing parameter to[2]

$$l^{-1} = 9cm\sqrt{k} \left(81 \times 8 \times 42\pi cm\sqrt{k}/\epsilon \right)^{1/(2m)}. \tag{27}$$

Using the proofs from [8] we can estimate the number of oracle calls that will be required to achieve the target error. Notice that, in these estimates, we consider only the asymptotic scaling (assuming remaining constant factors are 1) and assume that ϵ_{phase} is sufficiently small to have no impact on performance. This results in 6.3×10^7 oracle calls.

We can compare this asymptotically to the performance of both classical finite difference (with common random numbers and without) and semi-classical quantum finite difference. These results are summarized in Table 2. We choose the optimal values of the discretization step h as calculated in Section 2.2 to minimize the gradient estimation error ϵ, and pick $\epsilon = 10^{-3}$ for these benchmarks.

In Table 2, we notice that using the parameters from the proofs in [8], the GAW algorithm fails to deliver an advantage compared to the semi-classical method for k as high as 10^3. However, while this analysis gives us an idea of how the different methods compare in theory, the estimates are based on asymptotic bounds with parameters which may be loose in practice and are highly dependent on the smoothness of the functions considered. In the following section we study the performance of the GAW method numerically on small examples that are representative of some practical cases in finance.

---

[2]See proof of Theorem 5.4 from [8]

Classical Finite Difference	CFD with CRN	Semi-classical	GAW Quantum Gradient
10^{12}	10^{9}	10^{6}	10^{7}

Table 2: Table of asymptotic oracle calls required to compute quantum gradients for financially relevant greeks. Parameters for this benchmark are described in the text. This asymptotic analysis indicates that quantum gradient method has potential to outperform classical finite difference methods but is on par with semi-classical methods.

4.1 GAW Numerical estimates

The GAW method described in Sec. 3 gives a quantum algorithm for gradient estimation which scales as $\mathcal{O}(\sqrt{k}/\epsilon)$ when the target function satisfies the conditions of Theorem 1. However, in most relevant financial models of interest, we do not have access to closed-form solutions that we can examine to check whether they satisfy the smoothness conditions required of the theorem. As such, in this section we numerically examine the behavior of the high-order methods of the GAW gradient estimation algorithm for two financial use cases: (a) A simple (*vanilla*) European call option for which we have an analytical closed form solution and can benchmark the performance of the algorithm against the exact gradients, and (b) a path-dependent basket option with a *knock-in* feature which has no known analytical solution and is in practice classically evaluated using Monte Carlo methods. In particular, we examine the central-difference approximation order m and spacing l required for adequately precise gradient estimation and compare it to the theoretical values of Eq. (26) and Eq. (27) respectively. We focus on these two parameters because they determine the overall complexity of the algorithm, which we can then compare to the other methods of Table 2.

Because the resulting quantum circuits are prohibitively wide and deep for numerical simulation in practice, we adopt the following practical method to emulate the algorithm's performance: to estimate k greeks using n bits of precision we initialize a $k * 2^n$-dimensional array with the amplitudes of Eq. (18) computed classically for the chosen derivative order m. We then perform a $k * 2^n$-dimensional classical inverse Fourier transform of the array to get the resulting probability distribution which is the output of the GAW algorithm before measurement. In order to account for the phase error $\epsilon_{\mathrm{phase}} > 0$ from the Hamiltonian simulation, we add a random number to each encoded phase in Eq. (18), uniformly picked from the interval $[-\epsilon_{\mathrm{phase}}, \epsilon_{\mathrm{phase}}]$.

4.1.1 Vanilla Options

The simplest example in derivative pricing is a European call option whose price depends on the performance of a single asset at a pre-determined future time (the *expiration* date), relative to a reference price (the *strike*). In the Black-Scholes-Merton model [18], where the asset undergoes Geometric Brownian Motion (GBM), a call option on a non dividend-paying asset has a closed form solution given by

$$C = SN(d_1) - Ke^{-rT}N(d_2), \quad \text{with } d_1 = \frac{\ln(S/K) + (r + \sigma^2/2)T}{\sigma\sqrt{T}}, \ d_2 = d_1 - \sigma\sqrt{T}, \quad (28)$$

where S is the asset price today, K the strike of the option, r is a risk-free rate of return, σ the annualized volatility of the asset, T is the time until the option's expiration date and $N(x)$ denotes the CDF of the standard normal distribution. We test the quantum gradient estimation algorithm for the four greeks of this option/model: $delta = \partial C/\partial S$, $rho = \partial C/\partial r$, $vega = \partial C/\partial \sigma$, and $theta = \partial C/\partial T$. We numerically simulate the GAW algorithm for increasing k (the number of greeks we compute simultaneously) and central-difference approximation order $m \in [1, 4]$. In each case we search for the largest value of the spacing l in for which the algorithm produces an estimate ϵ-close to the exact value with probability $\geq 85\%$ for each greek. We target a gradient error of $\epsilon = 2 \times 10^{-2}$ which requires $n = \lceil \log(1/\epsilon) \rceil = 6$ qubits in each dimension. [3]

For each value of k, we then compare what we numerically find as the optimal values for (m, l) — that minimize the total number of oracle calls N_o from Eq. (25) while maintaining a

---

[3]Our numerical simulations scale exponentially in this parameter as are emulating the quantum circuit on these qubits. This limits us to this size.

success probability of $\geq 85\%$ — to those used in the proof of Theorem 1, given by Eq. (26) and Eq. (27), assuming the smoothest possible parameter $c = 1$. We set the approximation error from the Hamiltonian simulation to $\epsilon_{\text{phase}} = 10^{-4}$ and include it as an error source in our numerical simulations. The gradients of the vanilla option in Eq. (28) are evaluated at the point $(S, r, \sigma, T) = (99.5, 1\%, 20\%, 0.1)$, with $K = 100$, chosen so that the parameter values are reasonably realistic from a finance point of view, but at the same time probing a domain where the function is as non-linear as possible.[4] In Table 3 we show the results of the numerical simulation and the corresponding theoretical estimates from Theorem 1 and in Fig. 1 we show the resulting probability distribution from the quantum gradient estimation algorithm for each greek, for the case $k = 4$ in Table 3. In order to simulate the algorithm for increasing values of k, we pick k out of the four parameters S, r, σ, T with respect to which we compute the gradients and fix the values of the remaining $4 - k$ parameters. For the the cases $k \in [2, 3]$ where we have a choice of which parameters we fix, we have verified that the simulation results in Table 3 are qualitatively independent of the choice.

	Numerical			Theoretical		
k	m	l	N_o	m	l	N_o
2 (*delta, rho*)	1	0.65	1976	5	0.0028	570592
3 (*delta, rho, vega*)	3	0.65	4664	5	0.0022	712008
4 (*delta, rho, vega, theta*)	3	0.58	4904	5	0.0019	833296

Table 3: In this table we show (a) numerical estimates of the query complexity N_o (Eq. (25)) and parameter values (m, l) required by the GAW gradient estimation algorithm in order to estimate k greeks for the vanilla option of Eq. (28) within $\epsilon = 2 \times 10^{-2}$ with probability $\geq 85\%$ and (b) the corresponding values used in the proof of Theorem 1. We notice that for a vanilla option, we can reduce the central-difference order and increase the spacing by more than two orders of magnitude compared to the theoretical values. As such, the query complexity we estimate numerically is ~ 200 times smaller than the theoretical.

Interestingly, we notice from Table 3 that the query complexity required to estimate the gradients of the vanilla option with high probability in practice, is orders of magnitude smaller than what is expected from the parameters used in proof of Theorem 1. Because the types of vanilla options explored in this section are primarily a motivating example of relevance to finance that are simpler to analyze, we now turn our attention to more complex derivatives for which gradient estimation is required for business use in practice.

4.1.2 Path-dependent Basket Options

In the previous section, we numerically examined the performance of the GAW algorithm for vanilla options for which we have analytical solutions and we can benchmark the algorithm's performance compared to the exact gradients of the model. We now perform a similar analysis to a path-dependent option on multiple underlying assets, which has no known closed form solution, and compare the query complexity from the numerical simulation to the theoretical complexity from Theorem 1 as well as to that of the SQG method (Sec. 2.4) which calculates gradients using finite-difference using values estimated using QAE. Similarly to the previous section, we set the approximation error from the Hamiltonian simulation to $\epsilon_{\text{phase}} = 10^{-4}$.

The option we consider in this section is defined on three underlying assets undergoing GBM with volatilities $\sigma_1 = 20\%, \sigma_2 = 20\%, \sigma_3 = 10\%$ and spot prices $\vec{S}(t = 0) = (S_1(t = 0), S_2(t = 0), S_3(t = 0)) = (2.0, 2.0, 2.0)$. The risk free rate is set to $r = 1\%$ and the option expires in $T = 3$ years. The weighted sum of the asset prices $\vec{w} \cdot \vec{S}(t)$ with weights $\vec{w} = (w_1, w_2, w_3) = (0.5, 0.3, 0.2)$ is observed on five days $t^B = [T/5 * i]$ for $i \in [1, 5]$ across the duration of the contract and the

---

[4]Where the function is (approximately) linear then simple finite difference methods perform well already.

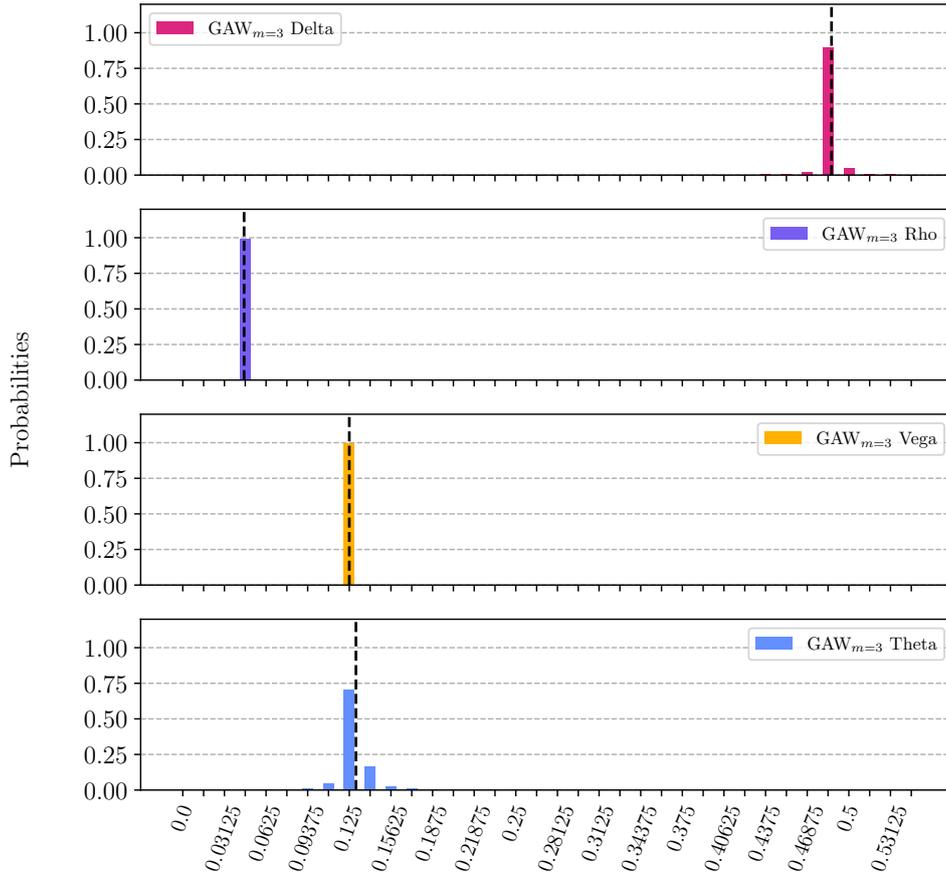Accepted in ⟨ Ӽuantum 2022-05-22, click title to verify. Published under CC-BY 4.0.
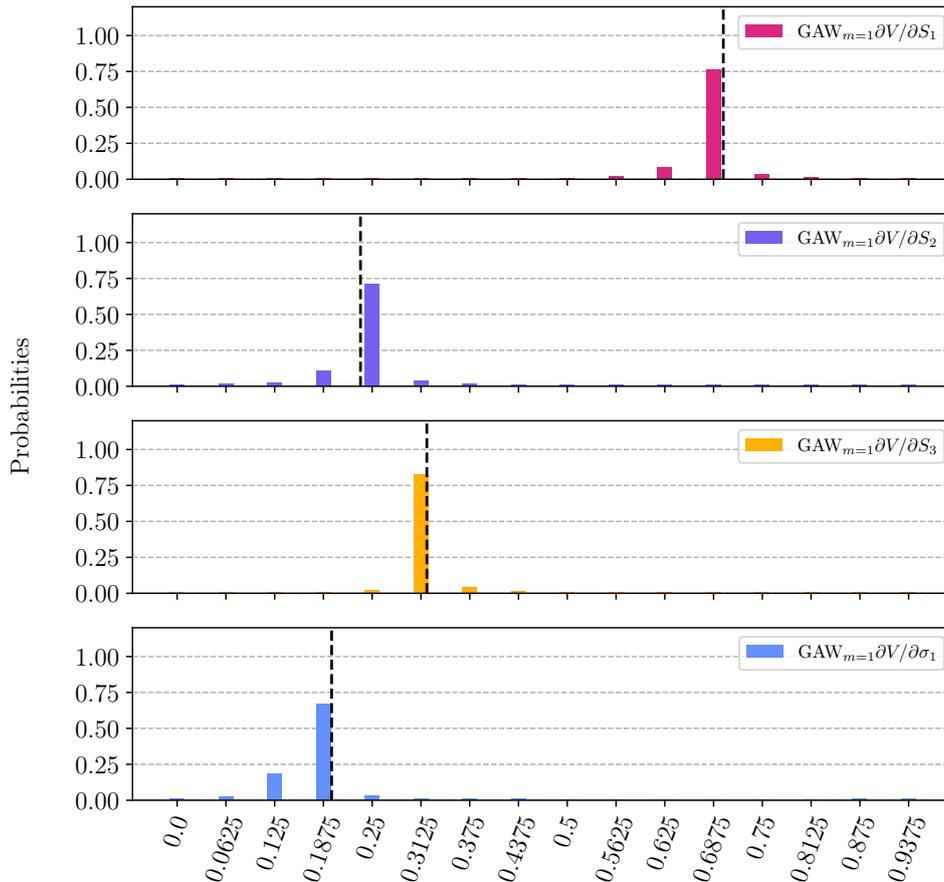
10

Figure 1: The probability distribution resulting from the numerical simulation of the GAW quantum gradient estimation algorithm with $m = 3$ and $l = 0.58$ for the four greeks (delta, rho, vega, theta) of the vanilla option of Eq. (28), and the corresponding exact values shown as dashed vertical lines. Measurement of the registers corresponding to each greek will result in a value at most $\epsilon = 1/N = 2 \times 10^{-2}$ away from the exact value with probability $\geq 85\%$.

option's payoff is given by

$$f(\vec{S}(t), K, B) = \begin{cases} \max(\vec{w} \cdot \vec{S}(T) - K, 0), & \text{if } \vec{w} \cdot \vec{S}(t) > B \text{ for any } t \in t^B \\ 0, & \text{otherwise,} \end{cases} \quad (29)$$

where we set the strike $K = 1.0$ and $B = 2.5$. This option is said to have a *knock-in* feature, because it only pays off (i.e. is *knocked-in*) if the observed weighted sum exceeds the knock-in barrier B at any of the pre-defined times before the contract's expiration. We simulate the GAW algorithm to estimate four gradients of this option contract's price V: the three deltas ($\partial V/\partial S_i$) and the vega with respect to the first asset $\partial V/\partial \sigma_1$. Because this option does not have a closed-form solution, we price the option using classical Monte Carlo with 10^6 paths and use finite-difference to compute the expected gradients which we use as benchmarks for the quantum algorithm. The quantum algorithm is simulated using $n = 4$ qubits for each gradient register, which sets the target error of the algorithm to $\epsilon \leq 1/2^4 = 0.0625$. Using $m = 1$ and $l = 0.25$ the simulated GAW algorithm gives us an ϵ-close estimate with probability $\geq 85\%$ for each greek. The SQG method requires $1/\epsilon$ calls to the $\mathcal{Q}$ operator of QAE, and each $\mathcal{Q}$ includes two calls to the unitary $\mathcal{A}$ of Eq. (1) and its inverse, therefore the complexity for k greeks is $2k/\epsilon$. Because with the SQG method we need to compute the payoff twice in order to construct the finite difference (Eq. (14)), the $\mathcal{A}$ operator will be approximately twice as large as the regular pricing oracle. In order to compare the query complexity more accurately with the other quantum methods we thus include a factor of two in the complexity, for a total of $4k/\epsilon$.

In Table 4 we show the query complexity and parameters from the numerical simulation of the GAW method for this path-dependent basket option, along with (a) the asymptotic estimates from Theorem 1, (b) the query complexity of the SQG method and (c) the query complexity of the CFD and CFD-CRN methods, all for the same target approximation error. For the CFD and CFD-CRN methods, the reported query complexity is the total number of Monte Carlo paths required for the evaluation of the k greeks within the target approximation error ϵ with probability $\geq 85\%$, computed numerically. The table also includes the parameters and resources required for the same calculation using the *Simulation-Free Quantum Gradient* (SFQG) algorithm described in the next section. The resulting probability distribution for all greeks along with the MC-estimated "true" values is shown in Fig. 2.

Figure 2: The probability distribution resulting from the numerical simulation of the GAW quantum gradient estimation algorithm with $m = 1$ and $l = 0.25$ for the four greeks of the path-dependent basket option of Sec. 4.1.2, along with the values estimated using classical Monte Carlo (MC) with 10^6 paths (vertical dashed lines). Measurement gives us estimates for each greek with error $\epsilon \leq 0.0625$ with probability $\geq 85\%$.

5 Simulation-Free Quantum Gradient Method

In this section, we construct a second-order accurate quantum gradient algorithm, corresponding to $m = 1$ in Eq. (17). This is higher order than Jordan's algorithm, which has unfavorable scaling, but lower order than required from the analysis in Gilyén et al. to guarantee the $\mathcal{O}(\sqrt{k}/\epsilon)$ scaling. A benefit is that, in the case of derivative pricing, we are able to give an explicit construction of the phase oracle without needing to appeal to block encoding or Hamiltonian simulation. We call this the *Simulation-Free Quantum Gradient* (SFQG) method.

To describe this construction, we first show how to turn the derivative pricing setting of Sec. 1.1 into a first-order phase oracle O_{Sf} and then how to build a second-order extension to construct

O_{Sf}^1 as defined in Eq. (18). We then simulate this algorithm to estimate the greeks of the basket option from Sec. 4.1.2, allowing us to compare its performance to the corresponding second-order accurate GAW method.

5.1 First-Order Pricing Phase Oracle

In order to apply the quantum gradient algorithms to estimate the gradient of a function $f(\mathbf{x})$, we need to construct a phase oracle of the form of Eq. (15). We evaluate this oracle on a superposition of points $\mathbf{x}$ in a k-dimensional hypercube of edge length l, $G^k = [-l/2, l/2]^k$ around $\mathbf{x}_0$, where each dimension is discretized using N points with $n = \log N$ qubits and l chosen small enough so that $f(\mathbf{x})$ is approximately linear in that region. For simplicity, we let $\mathbf{x}_0 = \mathbf{0}$ as gradient estimation at other points can be achieved by trivially redefining the function f. Evaluating the $\mathcal{A}$ operator of Eq. (1) on all points $\mathbf{x}$ in superposition we get

$$\frac{1}{\sqrt{N^k}} \sum_{\mathbf{x}} |\mathbf{x}\rangle |0\rangle_{q+1} \xrightarrow{\mathcal{A}} \frac{1}{\sqrt{N^k}} \sum_{\mathbf{x}} |\mathbf{x}\rangle \left(\sqrt{1 - f(\mathbf{x})} |\psi_0\rangle_q |0\rangle + \sqrt{f(\mathbf{x})} |\psi_1\rangle_q |1\rangle \right)$$

$$= \frac{1}{\sqrt{N^k}} \sum_{\mathbf{x}} |\mathbf{x}\rangle \left(|\Psi_+(\mathbf{x})\rangle - |\Psi_-(\mathbf{x})\rangle \right) \tag{30}$$

where $|\Psi_\pm(\mathbf{x})\rangle = -i e^{\pm i\theta(\mathbf{x})} |\psi_\pm(\mathbf{x})\rangle / \sqrt{2}$ and $f(\mathbf{x}) = \sin^2(\theta(\mathbf{x}))$, similarly to Eq. (3). Now, define the Grover operator $\mathcal{Q} = \mathcal{A} S_0 \mathcal{A}^\dagger S_{\psi_0}$, where $S_0 = \mathbb{I}^{\otimes nk} \otimes (\mathbb{I} - 2|0\rangle_{q+1}\langle 0|_{q+1})$ and $S_{\psi_0} = \mathbb{I}^{\otimes nk} \otimes (\mathbb{I} - 2|\psi_0\rangle_q|0\rangle\langle 0|\langle\psi_0|_q)$. Let D be an upper bound on $|(\partial\theta(\mathbf{x})/\partial x_i)|$ for all $i \in [1, k]$ and apply the $\mathcal{Q}$ operator πS times to the state in Eq. (30) with $S = N/Dl$ to get [5]

$$\mathcal{Q}^{\pi S} \mathcal{A} : \frac{1}{\sqrt{N^k}} \sum_{\mathbf{x}} |\mathbf{x}\rangle |0\rangle_{q+1} \to \frac{1}{\sqrt{N^k}} \sum_{\mathbf{x}} e^{2\pi i S\theta(\mathbf{x})} |\mathbf{x}\rangle |\Psi_+(\mathbf{x})\rangle - \frac{1}{\sqrt{N^k}} \sum_{\mathbf{x}} e^{-2\pi i S\theta(\mathbf{x})} |\mathbf{x}\rangle |\Psi_-(\mathbf{x})\rangle . \tag{31}$$

Note that the application of $O_{S\theta} \equiv \mathcal{Q}^{\pi S} \mathcal{A}$ in Eq. (31) is close to the phase oracle we are looking for in quantum gradient estimation, but (i) for θ rather than f, and (ii) with a superposition over the positive and negative phases that we seek.

An inverse Quantum Fourier Transform $\left(\mathcal{F}_n^{-1} \right)^{\otimes k}$ on the first register then gives us an estimate of the derivatives of $\theta(\mathbf{x})$ at $\mathbf{x}_0$

$$\left| \frac{N}{D} \frac{\partial\theta}{\partial x_1} \right\rangle \left| \frac{N}{D} \frac{\partial\theta}{\partial x_2} \right\rangle \cdots \left| \frac{N}{D} \frac{\partial\theta}{\partial x_k} \right\rangle \quad \text{or} \quad \left| -\frac{N}{D} \frac{\partial\theta}{\partial x_1} \right\rangle \left| -\frac{N}{D} \frac{\partial\theta}{\partial x_2} \right\rangle \cdots \left| -\frac{N}{D} \frac{\partial\theta}{\partial x_k} \right\rangle . \tag{32}$$

This method gives us an oracle for θ and not for f directly. However, because we know that θ and f are related through $f(\mathbf{x}) = \sin^2(\theta(\mathbf{x}))$, we can compute the derivatives of θ and then use the chain rule to get the gradients of f:

$$\pm \frac{\partial f}{\partial x_i}\bigg|_{\mathbf{x}_0} = \pm \frac{\partial\theta}{\partial x_i}\bigg|_{\mathbf{x}_0} \times \sin(2\theta(\mathbf{x}_0)) . \tag{33}$$

This requires knowledge of $\theta(\mathbf{x}_0)$ which can be calculated separately through standard QAE via the operator of Eq. (1).

The positive and negative cases can be distinguished by adding a dummy dimension to f with a gradient that has a known sign. For example one could transform the function $\left(f : \mathbb{R}^k \mapsto \mathbb{R} \right) \mapsto \left(f + 0.5 x_{k+1} : \mathbb{R}^{k+1} \mapsto \mathbb{R} \right)$. Inspecting the sign of the $k+1$-th derivative tells us if we are in the positive or negative case.

Because in this setting the value of $\pi S = \pi N/Dl$ indicates the number of times we need to invoke the $\mathcal{Q}$ operator in Eq. (31), it must be expressible as an integer. Since $N = 2^n$ is an integer, this can be achieved by picking a value of D such that π/Dl is also an integer. After we apply the $\mathcal{Q}$ operator the resulting integral number of times, the measured gradients in Eq. (32) are classically multiplied by the choice of D in order to recover the correct magnitude of the gradients.

---

[5] Because the states $|\Psi_\pm(\mathbf{x})\rangle$ in Eq.(31) depend on $\mathbf{x}$, we must be careful to make sure they do not interfere with the phase kicked back to the $|\mathbf{x}\rangle$ register. As we discuss in Appendix A the structure of the $\mathcal{A}$ operator used in derivative pricing allows the phase kickback to take place correctly.

5.2 Second-Order Pricing Phase Oracle

The simplest high-order extension we can do is to construct an oracle which encodes the two-point approximation shown in Eq. (19), which corresponds to $m = 1$ in Eq. (17). Because in the QAE setting we are computing the gradients of $\theta(\mathbf{x}) = \sin^{-1}\sqrt{f(\mathbf{x})}$ instead of $f(\mathbf{x})$, we need to construct an oracle which performs

$$O^1_{S\theta}\,|\mathbf{x}\rangle = e^{2\pi i S(\theta(\mathbf{x}) - \theta(-\mathbf{x}))/2}\,|\mathbf{x}\rangle, \tag{34}$$

where $\theta(-\mathbf{x}) = \sin^{-1}\sqrt{f(-\mathbf{x})}$. In [8], the authors suggest that high-order oracles of this form can be constructed as the product of two separate oracles with opposite phases. The same idea can be used to construct the oracle of Eq. (34) using appropriately defined oracles. Let $\mathcal{A}_+$ label the $\mathcal{A}$ operator of Eq. (1) and define operator $\mathcal{A}_-$ which acts as a probability oracle for the value $f(-\mathbf{x})$

$$\mathcal{A}_- : |\vec{0}\rangle\,|\mathbf{x}\rangle \to \left(\sqrt{1 - f(-\mathbf{x})}\,|\psi_0(-\mathbf{x})\rangle\,|0\rangle + \sqrt{f(-\mathbf{x})}\,|\psi_1(-\mathbf{x})\rangle\,|1\rangle\right)|\mathbf{x}\rangle, \tag{35}$$

as well as corresponding Grover operators $\mathcal{Q}_+ = \mathcal{A}_+ S_0 \mathcal{A}_+^\dagger S_{\psi_0}$ and $\mathcal{Q}_- = \mathcal{A}_- S_0 \mathcal{A}_-^\dagger S_{\psi_0}$. From Eq. (31) we see that the product of the oracles $O^+_{S\theta} \equiv \mathcal{Q}^{\pi S}_+ \mathcal{A}_+$ and $O^-_{S\theta} \equiv \mathcal{Q}^{\pi S}_- \mathcal{A}_-$ generates the state

$$\frac{1}{\sqrt{N^k}}\sum_{\mathbf{x}} e^{2\pi i S(\theta(\mathbf{x}) - \theta(-\mathbf{x}))}\,|\mathbf{x}\rangle\,|\psi_1(\mathbf{x})\rangle + \frac{1}{\sqrt{N^k}}\sum_{\mathbf{x}} e^{-2\pi i S(\theta(\mathbf{x}) - \theta(-\mathbf{x}))}\,|\mathbf{x}\rangle\,|\psi_2(\mathbf{x})\rangle$$

$$+ \frac{1}{\sqrt{N^k}}\sum_{\mathbf{x}} e^{2\pi i S(\theta(\mathbf{x}) + \theta(-\mathbf{x}))}\,|\mathbf{x}\rangle\,|\psi_3(\mathbf{x})\rangle + \frac{1}{\sqrt{N^k}}\sum_{\mathbf{x}} e^{-2\pi i S(\theta(\mathbf{x}) + \theta(-\mathbf{x}))}\,|\mathbf{x}\rangle\,|\psi_4(\mathbf{x})\rangle \tag{36}$$

where ψ_i denotes products of eigenstates of $\mathcal{Q}_+$ and $\mathcal{Q}_-$ which can then be ignored for the rest of the algorithm. While the first two terms contain the appropriate phase kickback (up to the sign) for the two-point approximation method of Eq. (34) with combined probability of 50%, the last two terms encode a phase proportional to $\theta(\mathbf{x}) + \theta(-\mathbf{x}) = \theta(\mathbf{x}_0) + \mathcal{O}(\partial^2\theta(\mathbf{x})/\partial\mathbf{x}^2)$, which create a probability peak around zero instead of the gradient, with similar combined 50% probability. By adding a dummy variable to the function as described at the end of Sec. 5.1, we can distinguish which eigenstate we are in by measuring the gradient of the dummy variable after applying the inverse Quantum Fourier Transform. Because we know the gradient with respect to the dummy variable by construction, measuring the positive (negative) gradient of the dummy variable means we are measuring the positive (negative) gradient with respect to the other variables. Otherwise, if we measure zero in the dummy variable register, we ignore that measurement. This means that additional post-processing is required for this method, and 50% of the shots are discarded. The total number of shots required depends on the desired accuracy of the estimation and this choice is discussed in more detail in Sec. 6. A circuit diagram of the SFQG method is shown in Fig. 3. Note that the operators $\mathcal{Q}_+$ and $\mathcal{Q}_-$ need to be applied to separate registers in order to generate the correct phases for Eq. (36).

Using the operator to construct the state of Eq. (36), we numerically simulate the SFQG algorithm to compute the greeks of the basket option of Sec. 4.1.2. In this case we use the algorithm to compute the gradients $\partial\theta/\partial S_1$, $\partial\theta/\partial S_2$, $\partial\theta/\partial S_3$, $\partial\theta/\partial\sigma_1$, where $\theta = \sin^{-1}\sqrt{V}$, and V is the option price, from which we can then estimate the option's greeks using Eq. (33), assuming we have already priced the contract. Similarly to the simulation of the GAW algorithm in Sec. 4.1.2, we search for the parameter value l which estimates the gradients within $\epsilon \leq 0.0625$ with probability of success $\geq 85\%$, using $n = 4$ qubits for each gradient register. Here we ignore the probability of obtaining the states we discard in Eq. (36) in post-processing. Intuitively, we can think of the effective oracular cost as being twice what we compute from this simulation given that we have 50% probability of failure, but as we show in Sec. 6 there is a more efficient way of combining measurement results to obtain estimates and confidence intervals for the gradients. The complexity of the algorithm in terms of the number of serial invocations to the $\mathcal{A}$ operator of Eq. (1) required to construct the state of Eq. (36) is then $\pi S = \pi N/l$, where $N = 2^n$.

The probability distribution in the non-discarded states for all four gradients after the application of the SFQG algorithm is shown in Fig. 4, and the parameter values and query complexity of the algorithm is shown in Table 4 along with all the other quantum, classical and semi-classical methods studied in this manuscript. The classical method complexity is estimated by computing

the greeks with finite-difference using 3000 Monte Carlo simulations of the basket option price and searching for the number of Monte Carlo paths which give the same target error $\epsilon \leq 0.0625$ with probability $\geq 85\%$ as the quantum and semi-classical methods.
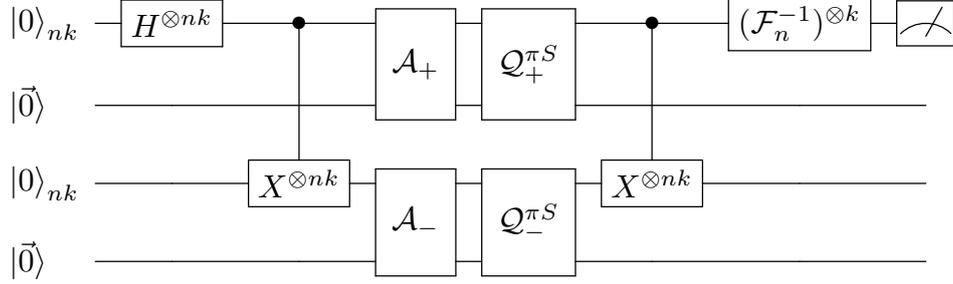

Figure 3: Circuit diagram of the SFQG method used to generate the state of Eq. (36). We first apply Hadamard gates to k registers of n qubits each to generate the superposition of points $\sum_{\mathbf{x}} |\mathbf{x}\rangle$. We then use CNOT gates to create a copy of each $|\mathbf{x}\rangle$ which allows us to evaluate the oracles $O^+_{S\theta} \equiv \mathcal{Q}^{\pi S}_+ \mathcal{A}_+$ and $O^-_{S\theta} \equiv \mathcal{Q}^{\pi S}_- \mathcal{A}_-$ in parallel at the cost of extra qubits. After uncomputing the copies, an n-dimensional inverse Quantum Fourier Transform in each of the initial k registers gives us the state of Eq. (36) which we then measure to estimate the k gradients.

		N_o	m	l
Quantum	Simulation-Free (SFQG)*	201	1	0.25
	Semi-classical (SQG)	256	n/a	n/a
	GAW (Numerical)	1600	1	0.25
	GAW (Theoretical)	201,528	4	0.0018
Classical	Finite-Difference w/ CRN (CFD-CRN)	32,000	n/a	n/a
	Finite-Difference (CFD)	400,000	n/a	n/a

Table 4: Comparison between (a) the numerical estimates of the query complexity N_o (Eq. (25)) and parameter values (m, l) required by the GAW gradient estimation algorithm in order to estimate k greeks for the basket option of Eq. (29) within $\epsilon \leq 0.0625$ with probability $\geq 85\%$, (b) the corresponding theoretical values used in the proof of Theorem 1, (c) the parameters and resources required for the same calculation using the Simulation-Free Quantum Gradient (SFQG) algorithm described in Sec. 5, (d) the query complexity of an SQG method for the same target accuracy and confidence interval and (e) the total number of simulated classical Monte Carlo paths required for the same accuracy and confidence interval using CFD and CFD-CRN methods. We numerically find that for this path-dependent basket option, the $m = 1$ GAW method can estimate the four greeks within ϵ with probability $\geq 85\%$, with ~ 125 times smaller query complexity implied by the proof of Theorem 1 and 20 times smaller than the best finite-difference-based classical method (CFD-CRN). (* For the SFQG method, we can think of the effective oracular cost as being twice what is reported in this table given that we have 50% probability of failure, but as we show in Sec. 6 there is a more efficient way of combining measurement results to obtain estimates and confidence intervals.)

6 Quantum Gradient Estimation using Maximum Likelihood Estimation

The success of the quantum gradient estimation algorithms studied so far in this manuscript, as well as those in [8, 9] is judged by their ability to estimate the gradient within ϵ with high probability. In [8] the authors suggest repeating the gradient estimation algorithm $\mathcal{O}(\log(k/\rho))$
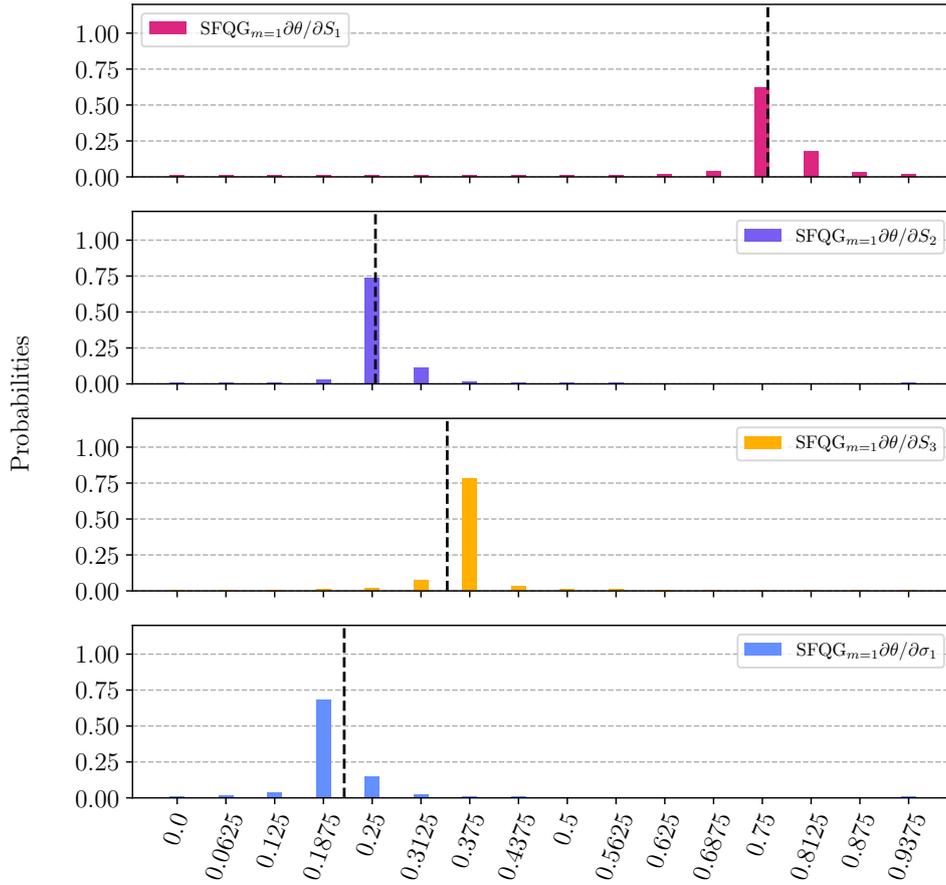
Figure 4: The probability distribution resulting from the numerical simulation of the second-order accurate, Simulation-Free Quantum Gradient (SFQG) algorithm described in Sec. 5 with $m = 1$ and $l = 0.25$ for the four greeks of the path-dependent basket option of Sec. 4.1.2, along with the exact values estimated using classical Monte Carlo simulation (vertical dashed lines). This algorithm allows us to estimate the gradients of the function $\theta = \sin^{-1}\sqrt{V}$, where V is the option price, from which we can then estimate the option's greeks using Eq. (33). Similarly to Fig. 2, measurement gives us estimates for each greek within error $\epsilon \leq 0.0625$ with probability $\geq 85\%$.

times for k gradients and taking the median to get the estimates within error ϵ with probability at least $(1 - \rho)$. However, this approach has two main drawbacks: a) In practice we would like to be able to characterize the correctness of the output with precise confidence intervals, and b) the output can only be one of the N possible values of the discretized hypercube $G_{\mathbf{x}_0}^k$ of Eq. (16).

Recently, approaches using Maximum Likelihood Estimation (MLE) have been proposed to address these issues for amplitude estimation algorithms [19–21], where the quantum circuits are sampled more than once, and the results are classically post-processed with MLE. In this section we show that we can apply the MLE method from [21] to the quantum gradient estimation algorithms to enhance the final estimate.

Given a probability distribution p with unknown parameter g and data x_i with $i = 1, \ldots, M$ sampled from it, MLE is used to obtain an estimate $\hat{g}$ for g. This is done by maximizing the log-likelihood $\log L$

$$\hat{g} = \arg\max_{g'} \log L(g') = \arg\max_{g'} \log \left(\prod_{i=1}^{M} p(x_i|g') \right) = \arg\max_{g'} \left(\sum_{i=1}^{M} \log p(x_i|g') \right), \qquad (37)$$

which measures how likely it is to measure the data x_i if g' is the true parameter. The general quantum gradient estimation algorithm from Sec. 2.5 requires a region l where the function whose gradients we are computing is approximately linear. In this case, the probability distribution

after the inverse Quantum Fourier Transform is applied to Eq. (16) is the same as that of phase estimation and is given by [11]

$$p(x) = \frac{\sin^2(N\Delta\pi)}{N^2\sin^2(\Delta\pi)}, \tag{38}$$

where $\Delta = (x - g')$ and N is the number of possible measurements.

Confidence intervals for the MLE estimate $\hat{g}$ can be derived using the likelihood ratio (LR) [22]. Following the analysis in the supplementary information of [21], the confidence interval at the $(1-\alpha)$ confidence level is the value g' satisfying $\left\{g' \in [0,1] : \log L(g') \geq \log L(\hat{g}) - q_{\chi_1^2}(1-\alpha)/2\right\}$, where q_{χ^2} denotes the $(1-\alpha)$ quantile of the χ^2 distribution.

In Fig. 5 we show how MLE can be used to improve the estimate of Vega $(\partial\theta/\partial\sigma)$ for the basket option from Fig. 4 calculated with the simulation-free quantum gradient method. As discussed in Sec. 5.2, the SFQG method generates a probability peak of 50% around the gradient up to the sign and another peak of 50% around zero which we need to ignore. Because we can distinguish between these cases by adding a dummy variable to the function with a known gradient, we can consider only the case where we measure the gradient, and double the number of required shots to take into account the discarded measurements. First, in Fig. 5a we show that the final probability distribution after the application of the quantum gradient estimation algorithm does indeed fit Eq. (38), and thus the QAE with MLE results from [21] can be used here too. For clarity, we only plot the measurement outcomes in the interval which contains most of the probability mass. In Fig. 5b we plot the log-likelihood $\log L(g')$ as a function of g' across the same interval when we sample the quantum gradient circuit that produces the probability distribution in Fig. 5a 30 times. The MLE estimate $\hat{g}$ is the global maximum of the function (green dot), which is very close to the true value g (dashed red line) and significantly better than the median of the final probability distribution (yellow arrow). In practice, the SFQG method requires double the number of shots (60) to account for the discarded shots due to the extra terms of Eq. (36).

We note that one significant advantage of the MLE method is that because the final estimate is not constrained to be one of the N possible discrete values, we can decrease the value of N at the cost of increasing the number of samples we take, thus lowering the overall width and depth of the quantum circuit. For instance, while the distance between the N possible discrete values in Fig. 5a is 0.0625, the error ϵ in the MLE estimate in Fig. 5b is $\epsilon = 4 \times 10^{-3}$.

While the MLE post-processing requires additional classical compute cost, calculating the MLE in this setting is done by maximizing a concave function over a one-dimensional compact interval, which we ignore in the overall complexity analysis. For more details on this process, we refer the reader to the supplementary information of [21].

7 Updated Estimates for Quantum Advantage

The resource estimation of Chakrabarti et al. [5] established that quantum advantage for derivative pricing with respect to classical Monte Carlo methods could require a quantum processor that can execute $\sim 10^7$ T-gates per second at a code distance that can support $\sim 10^{10}$ logical operations. More specifically, pricing the autocallable contract studied in Ref. [5] using the reparameterization technique introduced therein to within $\epsilon \leq 2 \times 10^{-3}$ with confidence level $1 - \alpha = 0.68$ requires a T-depth of 5×10^7. As such, in order to match the classical Monte Carlo pricing time estimated as 1 second, a logical quantum clock rate of 50MHz would be needed. We use the same reparameterization technique from Ref. [5] to construct oracle size estimates, and estimate the T-depth of the SFQG quantum gradient circuit required to compute the four greeks of the basket option of Sec. 4.1.2 and Table 4. We find that in order to calculate the greeks to within the same $\epsilon \leq 2 \times 10^{-3}$ and confidence level $1 - \alpha = 0.68$, we can use the SFQG gradient estimation method with the parameters of Fig. 4 and the maximum likelihood estimation (MLE) method from Sec. 6 using 60 shots (see Fig. 5b). Using these parameters, we estimate that the total T-depth of the SFQG circuit multiplied by the required number of shots is 5.5×10^7, the end-to-end circuit would require 12k logical qubits, and we would need to execute T-gates at a code distance that can support 10^8 logical operations. Assuming the contract can be classically priced using Monte Carlo in 1 second to within the same error and confidence level [5] and that the greeks can be calculated using
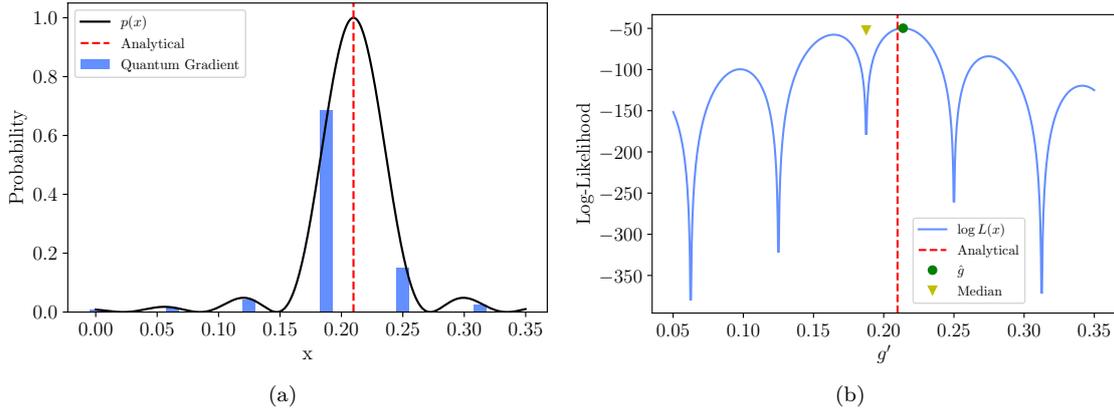
Figure 5: a) The discrete probability distribution before measurement of Vega ($\partial\theta/\partial\sigma$) for the basket option from Fig. 4 (blue bars) is fitted to the theoretical distribution $p(x)$ of Eq.(38) (black line). The maximum of the distribution occurs at the exact analytical value (dashed red line). b) The global maximum of the log-likelihood $\log L(g')$ (green dot) gives us a better estimate of the true value (dashed red line) than the most likely result if we sample from the probability distribution and take the median (yellow triangle). The log-likelihood plot was produced by sampling the quantum gradient estimation circuit 30 times and allows us to estimate this greek to within $\epsilon \leq 2 \times 10^{-3}$ with confidence level $1 - \alpha = 0.68$. When using the SFQG method we have 50% probability of discarding a measurement, and the expected number of shots for this target accuracy is thus 60.

a second-order finite-difference method applied to Monte Carlo pricings, the greeks of the basket option can be classically estimated in 8 seconds (four greeks, two pricings per greek). Therefore, quantum advantage in calculating the greeks of this derivative contract would require executing T-gates at a rate of 7MHz, ~ 7 times lower than the estimate of Chakrabarti et al. for quantum advantage in derivative pricing. Moreover, because the quantum gradient circuit is sampled 60 times when we use the MLE method (see Fig. 5), if we have parallel access to 60 QPUs on which the quantum circuit can be loaded and sampled simultaneously, we can achieve the same runtime as the serial execution if the logical cock rate of each device is ~ 100kHz, closer to current estimates of feasible logical clock rates around 10kHz [23].

While the computational cost required for the numerical simulations of the quantum gradient estimation algorithm limits our analysis to a maximum of $k = 4$ greeks for practically relevant use cases, we expect that the algorithm can scale favorably to derivative pricing problems of higher dimensionality, motivated by the fact that most, if not all, derivative contracts of practical interest have piecewise-linear payoffs [7]. While Theorem 1 provides a complexity of $\mathcal{O}(\sqrt{k}/\epsilon)$ for the class of functions considered, it does not preclude higher-order speedups with respect to k for smoother functions. For instance, the same gradient estimation algorithm applied to simple polynomial functions can achieve a complexity of $\mathcal{O}(\log(k)/\epsilon)$ [8]. Derivative contracts often include market parameters with little or no cross-dependence, i.e. $\partial^n f/(\partial x_1^m \partial x_2^{n-m}) \sim 0$ for $n > 1, m \in [1, n-1]$. For example, the basket option from Sec. 4.1.2 without the knock-in feature satisfies $\partial^n V/(\partial S_i^n \partial S_j^{n-m}) = 0$. The absence of such higher-order terms allows us to pick values of m and l in the application of the GAW algorithm which lead to smaller overall oracular cost than what is required by Theorem 1. An upper bound on the potential advantage compared to classical Monte Carlo is the case where the function is at most a second-degree polynomial in all k variables, yielding an overall speedup of $\mathcal{O}(k)$. Therefore, while we have made a first step in establishing the relevance and promise of the quantum gradient estimation methods in the context of financial derivative risk analysis, the possible extent of quantum speedup will be highly dependent on the nature of the price function for each derivative.

8 Discussion

We introduce a method to compute gradients of financial derivatives (greeks) using the gradient algorithms from [8, 9]. This method suggests additional quantum advantage is possible in risk

analysis, on top of the quadratic speedup of derivative pricing [1, 4, 5]. Classically computing k greeks with finite-difference methods - when the underlying derivative is priced using Monte Carlo - has complexity $\mathcal{O}(k/\epsilon^2)$ and straightforward extension of finite-difference methods to derivative pricing using amplitude estimation provides a quadratic advantage with complexity $\mathcal{O}(k/\epsilon)$. In this work, we explore an additional quadratic advantage for overall complexity of $\mathcal{O}(\sqrt{k}/\epsilon)$. The gradient estimation algorithm from [8] guarantees this quadratic advantage with respect to the number of greeks when the pricing function satisfies the smoothness conditions of Theorem 1. Because derivative pricing problems of practical interest in finance involve numerous diverse multivariate price functions and generally have no analytical solutions, understanding whether and which financial derivatives satisfy the aforementioned smoothness conditions is a challenging task. For this reason, we employ numerical methods to simulate the gradient estimation algorithm for two example derivatives: a) a European call option which has a closed-form solution and is used to establish the validity and benchmarks of the algorithm and b) a path-dependent basket option which has no known analytical solution and is representative of typical derivative price functions. We find that the quantum gradient algorithms not only succeed in estimating the associated greeks for these examples with high probability, but that the resulting query complexity is in fact significantly smaller than that suggested by Theorem 1 (Tables 3 and 4), suggesting that the associated price functions are smoother than those studied in Theorem 1.

Another question we tackle in this work is the rigorous resource estimation of the quantum oracles involved in the quantum gradient estimation algorithm. Due to the extra cost associated with the block-encoding and Hamiltonian simulation required to approximately construct the phase oracle of Eq. (15) from the probability oracle of Eq. (1) used in derivative pricing, in Sec. 5.2 we develop a method to construct a cheaper, second-order ($m = 1$) phase oracle exactly by taking advantage of the structure of amplitude estimation. An interesting question is whether this method can be extended to construct phase oracles for higher-order ($m > 1$) gradient methods which would apply more generally to quantum gradient estimation problems.

In Sec. 6 we show that it is possible to enhance quantum gradient algorithms by employing maximum likelihood estimation (MLE), allowing us to determine the resources required to estimate gradients with precise confidence intervals and confidence levels. Using this MLE method, in Sec. 7 we estimate the resources required for quantum advantage in derivative market risk for typical use cases of practical interest. We find that employing quantum gradient methods in derivative pricing lowers the logical clock rate estimate for quantum advantage from Chakrabarti et al. [5] by a factor of 7.

While finite-difference methods are still used in practice to compute greeks, more recently classical automatic differentiation (AD) methods have been gathering considerable interest because of their ability to significantly reduce the associated computational costs, at the cost of increased memory footprint [24, 25]. In particular, the adjoint mode of automatic differentiation (AAD) in certain cases allows the computation of all k gradients of a scalar function f at a cost which is independent of k, meaning that the overall classical complexity cost in this case becomes $\mathcal{O}(\omega/\epsilon^2)$, for some constant ω depending on the function f [26]. While the complexity of the GAW quantum gradient estimation algorithm scales as $\mathcal{O}(\sqrt{k}/\epsilon)$ for the class of smooth functions in Theorem 1, in Sec. 4.1 we saw that for practical use cases in finance, the algorithm scales 100x-200x times better than theoretical estimate from Theorem 1 for a given ϵ (Tables 3 and 4). As such, depending on the practical scaling of the GAW algorithm to larger values of k for finance use cases, it is possible that it could also outperform the complexity of AAD methods. It is also interesting to consider whether a similar construct as that employed by AD can be applied in a quantum setting. In Appendix B we provide such a construct and show that in certain settings it can lead to similar performance profile as classical AD, in that the runtime of the algorithm is independent of the number of greeks at the expense of increased memory usage. A detailed comparison of the performance between the quantum gradient algorithms and AD methods is a worthy study on its own and it is left for future research.

Intuitively, the expensive part of quantum derivative pricing is extracting the result through quantum amplitude estimation. Therefore it is advantageous to perform additional calculations involving that value before reading it out. In this work we consider calculating gradients, but there are other associated risk metrics that are equally of practical interest, such as the computation of portfolio value-at-risk (VaR). Based on our results, we are therefore cautiously optimistic that

further quantum advantage in the derivative pricing subroutine can be leveraged at these higher levels of calculation and aggregation, and suggests an additional research path going forward.

Acknowledgments

References

[1] P. Rebentrost, B. Gupt, and T. R. Bromley, "Quantum computational finance: Monte carlo pricing of financial derivatives," Phys. Rev. A **98**, 022321 (2018).

[2] S. Woerner and D. J. Egger, "Quantum risk analysis," npj Quantum Information **5** (2019), 10.1038/s41534-019-0130-6.

[3] D. J. Egger, R. G. Gutierrez, J. C. Mestre, and S. Woerner, "Credit risk analysis using quantum computers," IEEE Transactions on Computers (2020), 10.1109/TC.2020.3038063.

[4] N. Stamatopoulos, D. J. Egger, Y. Sun, C. Zoufal, R. Iten, N. Shen, and S. Woerner, "Option pricing using quantum computers," Quantum **4**, 291 (2020).

[5] S. Chakrabarti, R. Krishnakumar, G. Mazzola, N. Stamatopoulos, S. Woerner, and W. J. Zeng, "A threshold for quantum advantage in derivative pricing," Quantum **5**, 463 (2021).

[6] A. Montanaro, "Quantum speedup of monte carlo methods," Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences **471** (2015), 10.1098/rspa.2015.0301.

[7] J. Hull, *Options, futures, and other derivatives*, 6th ed. (Pearson Prentice Hall, Upper Saddle River, NJ [u.a.], 2006).

[8] A. Gilyén, S. Arunachalam, and N. Wiebe, "Optimizing quantum optimization algorithms via faster quantum gradient computation," Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms , 1425–1444 (2019).

[9] S. P. Jordan, "Fast quantum algorithm for numerical gradient estimation," Physical Review Letters **95** (2005), 10.1103/physrevlett.95.050501.

[10] S. Chakrabarti, A. M. Childs, T. Li, and X. Wu, "Quantum algorithms and lower bounds for convex optimization," Quantum **4**, 221 (2020).

[11] G. Brassard, P. Hoyer, M. Mosca, and A. Tapp, "Quantum Amplitude Amplification and Estimation," Contemporary Mathematics **305** (2002), 10.1090/conm/305/05215.

[12] P. Glasserman and D. Yao, "Some guidelines and guarantees for common random numbers," Management Science **38**, 884 (1992).

[13] B. Fornberg, "Generation of finite difference formulas on arbitrarily spaced grids," Mathematics of Computation **51**, 699 (1988).

[14] M. Gevrey, "Sur la nature analytique des solutions des équations aux dérivées partielles. premier mémoire," Annales scientifiques de l'École Normale Supérieure **3e série, 35**, 129 (1918).

[15] G. H. Low and I. L. Chuang, "Hamiltonian simulation by qubitization," Quantum **3**, 163 (2019).

[16] A. Gilyén, Y. Su, G. H. Low, and N. Wiebe, "Quantum singular value transformation and beyond: exponential improvements for quantum matrix arithmetics," in *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing* (2019) pp. 193–204.

[17] J. M. Martyn, Y. Liu, Z. E. Chin, and I. L. Chuang, "Efficient fully-coherent hamiltonian simulation," (2021), 10.48550/arXiv.2110.11327.

[18] F. Black and M. Scholes, "The pricing of options and corporate liabilities," Journal of Political Economy **81**, 637 (1973).

[19] Y. Suzuki, S. Uno, R. Raymond, T. Tanaka, T. Onodera, and N. Yamamoto, "Amplitude estimation without phase estimation," Quantum Information Processing **19**, 75 (2020).

[20] T. Tanaka, Y. Suzuki, S. Uno, R. Raymond, T. Onodera, and N. Yamamoto, "Amplitude estimation via maximum likelihood on noisy quantum computer," Quantum Information Processing **20**, 293 (2021).

[21] D. Grinko, J. Gacon, C. Zoufal, and S. Woerner, "Iterative quantum amplitude estimation," npj Quantum Information **7** (2021), 10.1038/s41534-021-00379-1.

[22] K.-R. Koch, *Parameter Estimation and Hypothesis Testing in Linear Models* (Springer-Verlag Berlin Heidelberg, 1999).

[23] A. G. Fowler and C. Gidney, "Low overhead quantum computation using lattice surgery," (2019), 10.48550/arXiv.1808.06709.

[24] C. Homescu, "Adjoints and automatic (algorithmic) differentiation in computational finance," Risk Management eJournal (2011), 10.2139/ssrn.1828503.

[25] G. Pages, O. Pironneau, and G. Sall, "Vibrato and automatic differentiation for high order derivatives and sensitivities of financial options," Journal of Computational Finance **22** (2016), 10.21314/JCF.2018.350.

[26] L. Capriotti, "Fast greeks by algorithmic differentiation," J. Comput. Financ. **14** (2010), 10.2139/ssrn.1619626.

[27] L. Capriotti and M. Giles, "Fast correlation greeks by adjoint algorithmic differentiation," ERN: Simulation Methods (Topic) (2010), 10.2139/ssrn.1587822.

[28] C. H. Bennett, "Logical reversibility of computation," IBM Journal of Research and Development **17** (1973), 10.1147/rd.176.0525.

A Phase Kickback

In order for the Simulation-Free Quantum Gradient (SFQG) method described in Sec. 5 to work, care must be taken for the appropriate phase kickback to occur in Eq. (31), so that we get the correct gradient after the application of the inverse Quantum Fourier Transform. Eq. (31) includes the states $|\Psi_{\pm}(\mathbf{x})\rangle$ which in general will interfere with the subsequent inverse Quantum Fourier Transform through their dependence on $\mathbf{x}$. The application of the $\mathcal{Q}$ operator in the SFQG method creates the state

$$|\Psi\rangle = \frac{1}{\sqrt{N^k}}\sum_{\mathbf{x}} e^{2\pi i S\theta(\mathbf{x})}|\mathbf{x}\rangle|\Psi_+(\mathbf{x})\rangle - \frac{1}{\sqrt{N^k}}\sum_{\mathbf{x}} e^{-2\pi i S\theta(\mathbf{x})}|\mathbf{x}\rangle|\Psi_-(\mathbf{x})\rangle. \tag{39}$$

Applying the inverse Quantum Fourier Transform to the $|\mathbf{x}\rangle$ register, we get

$$\frac{1}{N^k}\sum_{\mathbf{x}}\sum_{\mathbf{y}} e^{2\pi i(S\theta(\mathbf{x})-\mathbf{x}\cdot\mathbf{y}/N)}|\mathbf{y}\rangle|\Psi_+(\mathbf{x})\rangle - \frac{1}{N^k}\sum_{\mathbf{x}}\sum_{\mathbf{y}} e^{-2\pi i(S\theta(\mathbf{x})+\mathbf{x}\cdot\mathbf{y}/N)}|\mathbf{y}\rangle|\Psi_-(\mathbf{x})\rangle. \tag{40}$$

The probability of measuring a value $|\mathbf{z}\rangle$ in the first register is given by

$$\frac{1}{N^{2k}}\left(\sum_{\mathbf{x}'}\sum_{\mathbf{y}'} e^{-2\pi i(S\theta(\mathbf{x}')-\mathbf{x}'\cdot\mathbf{y}'/N)}\langle\mathbf{y}'|\langle\Psi_+(\mathbf{x}')| - \sum_{\mathbf{x}'}\sum_{\mathbf{y}'} e^{2\pi i(S\theta(\mathbf{x}')+\mathbf{x}'\cdot\mathbf{y}'/N)}\langle\mathbf{y}'|\langle\Psi_-(\mathbf{x}')|\right)(|\mathbf{z}\rangle\otimes\mathbb{I})(\langle\mathbf{z}|\otimes\mathbb{I})$$

$$\left(\sum_{\mathbf{x}}\sum_{\mathbf{y}} e^{2\pi i(S\theta(\mathbf{x})-\mathbf{x}\cdot\mathbf{y}/N)}|\mathbf{y}\rangle|\Psi_+(\mathbf{x})\rangle - \sum_{\mathbf{x}}\sum_{\mathbf{y}} e^{-2\pi i(S\theta(\mathbf{x})+\mathbf{x}\cdot\mathbf{y}/N)}|\mathbf{y}\rangle|\Psi_-(\mathbf{x})\rangle\right)$$

$$= \frac{1}{N^{2k}}\Bigg(\sum_{\mathbf{x}'}\sum_{\mathbf{x}} e^{-2\pi i(S\theta(\mathbf{x}')-\mathbf{x}'\cdot\mathbf{z}/N)}e^{2\pi i(S\theta(\mathbf{x})-\mathbf{x}\cdot\mathbf{z}/N)}\langle\Psi_+(\mathbf{x}')|\Psi_+(\mathbf{x})\rangle$$

$$+ \sum_{\mathbf{x}'}\sum_{\mathbf{x}} e^{2\pi i(S\theta(\mathbf{x}')+\mathbf{x}'\cdot\mathbf{z}/N)}e^{-2\pi i(S\theta(\mathbf{x})+\mathbf{x}\cdot\mathbf{z}/N)}\langle\Psi_-(\mathbf{x}')|\Psi_-(\mathbf{x})\rangle$$

$$- \sum_{\mathbf{x}'}\sum_{\mathbf{x}} e^{-2\pi i(S\theta(\mathbf{x}')-\mathbf{x}'\cdot\mathbf{z}/N)}e^{-2\pi i(S\theta(\mathbf{x})+\mathbf{x}\cdot\mathbf{z}/N)}\langle\Psi_+(\mathbf{x}')|\Psi_-(\mathbf{x})\rangle$$

$$- \sum_{\mathbf{x}'}\sum_{\mathbf{x}} e^{2\pi i(S\theta(\mathbf{x}')+\mathbf{x}'\cdot\mathbf{z}/N)}e^{-2\pi i(S\theta(\mathbf{x})-\mathbf{x}\cdot\mathbf{z}/N)}\langle\Psi_-(\mathbf{x}')|\Psi_+(\mathbf{x})\rangle\Bigg) \tag{41}$$

In the derivative pricing context we consider in this manuscript, the $\mathcal{A}$ operator we consider implements the *re-parameterization* method from [5]. In this case, the $\mathcal{A}$ operator can be written as the product of two operators $\mathcal{G}$ and $\mathcal{F}$. The $\mathcal{G}$ operator loads standard normal distributions corresponding to the number of assets of the derivative contract and the timesteps used in the pricing

$$\mathcal{G} : |0\rangle_m \to \sum_i \sqrt{p_i} \, |i\rangle \,, \tag{42}$$

where the probabilities p_i are independent of any market parameters. Then, the $\mathcal{F}$ operator computes the payoff $g(i)$ using quantum arithmetic on $|i\rangle$, which is subsequently rotated into the amplitude of an ancilla qubit

$$\mathcal{F} : \sum_i \sqrt{p_i} \, |i\rangle \, |0\rangle_q \to \sum_i \sqrt{p_i} \, |i\rangle \, |g(i)\rangle \, (\sqrt{1 - g(i)} \, |0\rangle + \sqrt{g(i)} \, |1\rangle). \tag{43}$$

After the final rotation, the register $|g(i)\rangle$ can be uncomputed, so that the overall effect of the $\mathcal{A} = \mathcal{F}(\mathcal{G} \otimes \mathbb{I}^{\otimes q})$ operator can be written as

$$\mathcal{A} \, |0\rangle_{m+1} = \sum_i \sqrt{p_i} \, |i\rangle \, (\sqrt{1 - g(i)} \, |0\rangle + \sqrt{g(i)} \, |1\rangle), \tag{44}$$

where $g(i) \in [0, 1]$. When $\mathcal{A}$ is evaluated in superposition over a register $|\mathbf{x}\rangle$ representing tweaks to input market parameters as shown in Eq.(30), the resulting state becomes

$$\mathcal{A} : \frac{1}{\sqrt{N^k}} \sum_{\mathbf{x}} |\mathbf{x}\rangle \, |0\rangle_{m+1} \to \frac{1}{\sqrt{N^k}} \sum_{\mathbf{x}} |\mathbf{x}\rangle \sum_i \sqrt{p_i} \, |i\rangle \, (\sqrt{1 - g(i, \mathbf{x})} \, |0\rangle + \sqrt{g(i, \mathbf{x})} \, |1\rangle)$$

$$= \frac{1}{\sqrt{N^k}} \sum_{\mathbf{x}} |\mathbf{x}\rangle \left(e^{i\theta(\mathbf{x})} |\Psi_+(\mathbf{x})\rangle - e^{-i\theta(\mathbf{x})} |\Psi_-(\mathbf{x})\rangle \right), \tag{45}$$

with

$$|\Psi_\pm(\mathbf{x})\rangle = \frac{1}{\sqrt{2}} \left(\sum_i \sqrt{p_i} \, |i\rangle \left(-i \sqrt{\frac{g(i, \mathbf{x})}{\sum_i p_i g(i, \mathbf{x})}} |1\rangle \pm \sqrt{\frac{1 - g(i, \mathbf{x})}{\sum_i p_i (1 - g(i, \mathbf{x}))}} |0\rangle \right) \right), \tag{46}$$

and

$$\theta(\mathbf{x}) = \arcsin \left(\sqrt{\sum_i p_i g(i, \mathbf{x})} \right) \tag{47}$$

where k is the dimension of the gradient we are estimating, and $N = 2^n$, and n qubits are used for the superposition $\sum_{\mathbf{x}} |\mathbf{x}\rangle$ in each dimension. The gradient estimation algorithm requires that we choose $\mathbf{x} \ll 1$ so that the function $\theta(\mathbf{x})$ is approximately linear in the vicinity of $\mathbf{x} = \mathbf{0}$. In this regime, the $|\Psi_\pm\rangle$ states in Eq.(46) give $\langle \Psi_+(\mathbf{x}')|\Psi_-(\mathbf{x})\rangle = \langle \Psi_-(\mathbf{x}')|\Psi_+(\mathbf{x})\rangle \approx 0$ and $\langle \Psi_+(\mathbf{x}')|\Psi_+(\mathbf{x})\rangle = \langle \Psi_-(\mathbf{x}')|\Psi_-(\mathbf{x})\rangle \approx 1$. The probability of measuring a value $|z\rangle$ in Eq.(41) is then given by

$$\frac{1}{N^{2k}} \sum_{\mathbf{x}'} \sum_{\mathbf{x}} e^{-2\pi i (S\theta(\mathbf{x}') - \mathbf{x}' \cdot \mathbf{z}/N)} e^{2\pi i (S\theta(\mathbf{x}) - \mathbf{x} \cdot \mathbf{z}/N)} = \frac{1}{N^{2k}} \left| \sum_{\mathbf{x}} e^{2\pi i (S\theta(\mathbf{x}) - \mathbf{x} \cdot \mathbf{z}/N)} \right|^2, \tag{48}$$

similarly to standard quantum phase estimation [11]. Note that the application of the $\mathcal{Q}$ operator in Eq. (39) induces the correct phase kickback to the $|\mathbf{x}\rangle$ register because the $\mathbf{x}$-dependence in Eq.(46) is limited to the amplitude of the last qubit. If the probabilities p_i become dependent on $\mathbf{x}$ or other qubit registers remain entangled with $\mathbf{x}$, the phase kickback fails.

In Fig. 6 we show the simulated measurement outcomes after the creation of the state in Eq. (39) and the subsequent inverse Quantum Fourier Transform for an example when the $\mathcal{A}$ operator is in the form of Eq. (45) and $k = 1$. The probabilities p_i are taken from a normal

distribution with unit variance defined on three qubits ($m = 3$), normalized such that $\sum_i p_i = 1$, and $g(i, x) = \sin^2(b * i + 2x)$ for $b = 0.405$. After applying the Quantum Fourier Transform and measuring the $|x\rangle$ register, we get an estimate of the gradient $d\theta/dx$ (or $-d\theta/dx$ as described by Eq.(32)) at $x = 0$ where $\theta = \arcsin(\sqrt{a})$ and $a = \sum_i p_i \sin^2(b * i + 2x)$. For the parameter values we have chosen, $d\theta/dx \approx 0.253$ at $x = 0$. We use three qubits for the register $|x\rangle$ ($n = 3$) which allows us to resolve the gradient with accuracy $1/8 = 0.125$. The value of b was chosen for clarity, so that the resulting gradient is close to one of the values which can be represented exactly with three qubits. We picked $l = \pi/256$ and $D = 1$ and thus the $\mathcal{Q}$ operator is applied $S = N/Dl = 2048$ times. Because the $\mathcal{A}$ operator used in this simulation is in the form of Eq. (45), the correct phase is kicked back to the $|x\rangle$ register, and the inverse Quantum Fourier Transform generates two probability peaks around $\pm d\theta/dx$ where $d\theta/dx \approx 0.253$
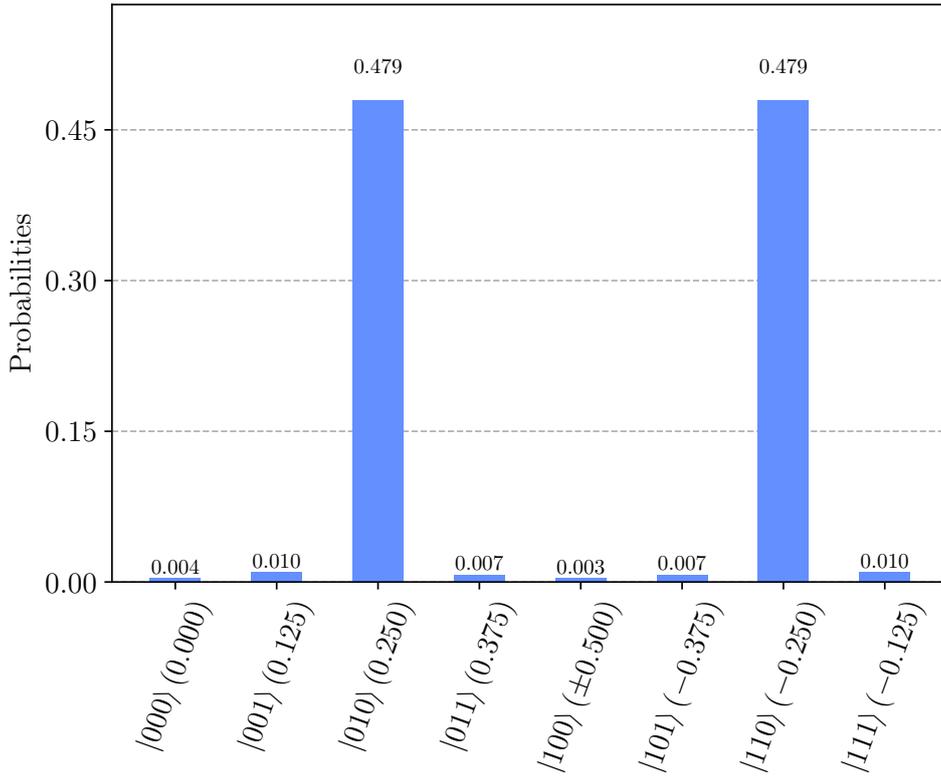

Figure 6: Simulated measurement outcomes after the application of Eq. (31) and the subsequent inverse Quantum Fourier Transform when the $\mathcal{A}$ operator is in the form of Eq. (45). The measurement outcomes correspond to estimates of the gradient $d\theta/dx$ at $x = 0$ where $\theta = \arcsin(\sqrt{a})$ and $a = \sum_i p_i \sin^2(b * i + 2x)$ for $b = 0.405$ and $p =$[0.03149738, 0.08388914, 0.16118575, 0.22342773, 0.22342773, 0.16118575, 0.08388914, 0.03149738]. The probability peaks around ± 0.25, which is the closest value representable with $n = 3$ qubits to the expected $d\theta/dx \approx 0.253$.

B   Automatic Differentiation and Multi-Objective QAE

Another way to compute gradients in the considered setting is automatic differentiation (AD) [24–26]. AD repeatedly applies the chain rule to every elementary arithmetic operation that is used to compute an objective function and keeps track of the analytical gradient throughout the calculation. Different variants of AD exist [24] and it has been shown that in many practical applications the gradient can be computed at only a constant overhead, independent of the dimension [26]. In some cases the overhead can even be stated to be bounded by a factor of four compared to evaluating the function alone at the expense of larger memory requirements [27].

Suppose now we want to estimate an expectation value $\mathbb{E}(g(S, x))$ for a payoff function g, a

random variable S and some given parameters x as well as the corresponding gradient $\nabla_x \mathbb{E}(g(S, x))$. To construct the probability oracle $\mathcal{A}(x)$ required by QAE for a fixed x, we usually first create a weighted superposition of all scenarios, then evaluate the corresponding payoff for each scenario, and last prepare an objective qubit, i.e., we get

$$\sum_{j=0}^{2^m-1} \sqrt{p_j} \, |s_j\rangle \, |g(s_j, x)\rangle \left(\sqrt{1 - g(s_j, x)} \, |0\rangle + \sqrt{g(s_j, x)} \, |1\rangle \right), \tag{49}$$

such that the probability of measuring $|1\rangle$ in the last qubit corresponds to $\mathbb{E}(g(S, x))$, and where the s_j denote the possible realizations of S represented by m qubits and the p_j denote the corresponding probabilities.

For every scenario $|s_j\rangle$, we apply quantum arithmetic to compute the payoff $|g(s_j, x)\rangle$. Thus, for each s_j, we can also use AD in the same way as classically to compute the gradient $\nabla_x g(s_j, x)$, while using at most twice the resources required classically due to the need of a reversible implementation [28]. Thus, with a constant overhead compared to the evaluation of the expectation value, this results in the state

$$\sum_{j=0}^{2^m-1} \sqrt{p_j} \, |s_j\rangle \, |g(s_j, x)\rangle \bigotimes_{i=1}^{k} |\partial_i g(s_j, x)\rangle. \tag{50}$$

In the following, we show how to use QAE to read out multiple objectives defined on the same random variables, which then immediately applies to the gradient as constructed in Eq. (50).

Suppose a random variable S and a set of functions f_i, $i = 1, \ldots, k$, that map realizations of S to $\mathbb{R}$. Further, suppose we are interested in estimating the expectation values $\mathbb{E}(f_i(S))$ for all i, and that we can construct a state of the form

$$\sum_{j=0}^{2^m-1} \sqrt{p_j} \, |s_j\rangle \bigotimes_{i=1}^{k} |f_i(s_j)\rangle. \tag{51}$$

Then, to estimate the values $\mathbb{E}(f_i(S))$, we first introduce k additional m-qubit registers $|c_i\rangle$, each initialized with some value c_i, and second, we use quantum arithmetic to compute the sum

$$\sum_{i=1}^{k} c_i f_i(s_j) \tag{52}$$

into another register. In other words, we construct an operator that acts as

$$\bigotimes_{i=1}^{k} |c_i\rangle \, |0\rangle \bigotimes_{i=1}^{k} |0\rangle \, |0\rangle \mapsto \bigotimes_{i=1}^{k} |c_i\rangle \sum_{j=0}^{2^m-1} \sqrt{p_j} \, |s_j\rangle \bigotimes_{i=1}^{k} |f_i(s_j)\rangle \, |\sum_{i=1}^{k} c_i f_i(s_j)\rangle. \tag{53}$$

By adding an objective qubit and applying a rotation controlled by the last register we can also use this to construct a probability oracle $\mathcal{A}(c)$ that corresponds to the function

$$f(c) \;=\; \mathbb{E}\left(\sum_{i=1}^{k} c_i f_i(S) \right). \tag{54}$$

This is a linear function in c and using the quantum gradient algorithm with respect to c results in

$$\nabla_c f(c) = (\mathbb{E}(f_1(S)), \ldots, \mathbb{E}(f_k(S)))^T, \tag{55}$$

i.e., in the read out of all k expectation values.

If the values of f_i are such that the weighted sum in Eq. (52) satisfies $\sum_{i=1}^{k} c_i f_i(s_j) \leq 1$, since the function by construction is linear in c, the resulting complexity of the quantum gradient algorithm for a target accuracy $\epsilon > 0$ scales as $\tilde{\mathcal{O}}(1/\epsilon)$, i.e., independent of k (ignoring logarithmic terms), following [8, Thm. 23, arxiv version]. The multi-objective QAE requires $k \cdot m$ additional qubits

as well as the weighted sum in Eq. (52), which can be computed in logarithmic depth by using a divide-and-conquer summation scheme. If on the other hand, Eq. (52) needs to be normalized by dividing the weighted sum with a factor D, the complexity of the algorithm becomes $\tilde{\mathcal{O}}(1/\epsilon D)$. When $f_i \sim 1, \forall i$, we would need to choose $D \sim 1/k$, which adds a factor of k back to the complexity of the algorithm, negating the advantage of this method.

Since Eq. (50) has the required shape, we can immediately apply the multi-objective QAE to evaluate the gradient that has been evaluated using AD implemented by quantum arithmetic. If no additional normalization is required, we can combine AD and (multi-objective) QAE to get the gradient algorithm with runtime $\tilde{\mathcal{O}}(1/\epsilon)$, i.e., independent of the dimension and with a quadratic speed-up in the accuracy. Thus, like classically, AD could represent a promising approach to estimate market risks with a significant advantage over finite difference schemes. What remains to be analyzed in more depth, is how this affects the required memory, i.e. qubits, and how to best automate automatic differentiation for reversible quantum arithmetic.