

Option pricing under stochastic volatility on a quantum computer

Guoming Wang and Angus Kan

ORCA Computing

We develop quantum algorithms for pricing Asian and barrier options under the Heston model, a popular stochastic volatility model, and estimate their costs, in terms of T-count, T-depth and number of logical qubits, on instances under typical market conditions. These algorithms are based on combining well-established numerical methods for stochastic differential equations and quantum amplitude estimation technique. In particular, we empirically show that, despite its simplicity, weak Euler method achieves the same level of accuracy as the better-known strong Euler method in this task. Furthermore, by eliminating the expensive procedure of preparing Gaussian states, the quantum algorithm based on weak Euler scheme achieves drastically better efficiency than the one based on strong Euler scheme. Our resource analysis suggests that option pricing under stochastic volatility is a promising application of quantum computers, and that our algorithms render the hardware requirement for reaching practical quantum advantage in financial applications less stringent than prior art.

1 Introduction

Recently, there has been rising interest in leveraging quantum technologies to tackle the computational problems in finance better than what is possible in the classical world (see [1] for a survey). In particular, multiple quantum algorithms [2, 3, 4, 5, 6, 7, 8, 9] have been proposed to price financial derivatives and assess financial risks faster than their classical counterparts. In this work, we focus on the pricing of options, i.e., derivative contracts that grant buyers the right, but not the obligation, to buy or sell an underlying asset at an agreed-upon price and date or within a specific time frame. Determining the fair market value of an option is of paramount importance in quantitative finance. Yet it could be difficult to solve precisely due to the stochastic nature of financial markets. Classical Monte Carlo methods are often employed to draw random paths from which one can estimate the discounted average payoff of the option at expiration [10, 11, 12]. By utilizing quantum techniques, especially amplitude estimation [13], one can achieve quadratic speedup in this process and thus obtain more accurate estimate of the target quantity in shorter time, provided we have sufficiently powerful quantum hardware.

Nevertheless, previous works on quantum algorithms for derivative pricing have been restricted to the basic Black-Scholes model [14, 15] which assumes that the asset price follows a geometric Brownian motion (GBM) with constant drift and volatility. Even though

Guoming Wang: guoming.wang.cs@gmail.com

Angus Kan: ckan@wesleyan.edu

this model is highly successful and easy to use, it has several limitations, including the assumption that the volatility of asset price remains constant over time. This assumption is unrealistic, because market volatility not only fluctuates over time but also appears to be random. To better reflect this reality, various stochastic volatility models have been proposed, and the *Heston model* [16] is one of the most popular among them. This model is described by two coupled stochastic differential equations (SDEs), one for the asset price and one for the asset volatility. Derivative pricing under the Heston model is more difficult than the one under the Black-Scholes model, as it often has no closed-form solutions and thus often involves expensive numerical simulation. In fact, finding efficient (classical) algorithms for this problem remains an active topic of research in (classical) computational finance until today [17, 18, 19, 20].

In this work, we initiate the study of option pricing under the Heston model on a quantum computer. Specifically, we develop quantum algorithms for pricing two common exotic options – Asian and barrier options – under this stochastic volatility model by combining classical numerical methods for SDEs [12] and quantum amplitude estimation. We first apply three numerical schemes, including strong Euler, weak Euler and order 2.0 Taylor methods, to this problem and empirically evaluate their performance. Here weak Euler method is a variant of the better-known strong Euler method (also known as the Euler-Maruyama method) in which Gaussian random variables are replaced with simpler discrete random variables. Surprisingly, despite its simplicity, weak Euler method achieves the same level of accuracy as the other two in our experiments. Furthermore, although order 2.0 Taylor method is significantly more complicated than strong and weak Euler methods, we do not witness any advantage of the former over the latter in our experiments.

In light of these findings, we then devise quantum algorithms for option pricing under the Heston model by combining strong and weak Euler schemes and iterative quantum amplitude estimation [21]. This enables us to achieve quadratic speedup in the estimation of the discounted average payoff over traditional Monte Carlo methods. Furthermore, to better understand the practicality of our algorithms, we explicitly construct the circuits for the unitary operations in these algorithms, and estimate the costs and errors of these algorithms on four example instances under typical market conditions. Here we assume a Clifford + T gate set, and use T-count, T-depth and the number of logical qubits in the circuit as our cost metrics, since T gates typically dominate the computational costs [22, 23]. In developing the algorithm based on strong Euler scheme, we also optimize the circuits of [24] for preparing quantum states encoding Gaussian distributions and significantly reduce their T-counts and T-depths, which might be of independent interest.

Under the settings delineated in Tables 3, 4 and 5, we obtain the resource estimates summarized in Table 6. They indicate that the quantum algorithm based on weak Euler scheme is far more efficient than the one based on strong Euler scheme, mainly because it avoids the expensive procedure of preparing Gaussian states. Furthermore, our resource analysis suggests that option pricing under stochastic volatility could be a promising application of quantum computers, and the hardware requirement for reaching quantum advantage using our algorithms is less stringent than previous ones for similar tasks.

The remainder of the paper is organized as follows. In Section 2, we review the basics of mathematical finance and numerical methods for SDEs which are necessary for understanding this work. Then in Section 3, we apply three numerical methods to option pricing under the Heston model and empirically evaluate their performance. Next, in Section 4, we develop our quantum algorithms for the same problem based on strong and weak Euler schemes and estimate their costs and errors on four example instances. Finally, Section 5 concludes this paper and points out directions for future work.

2 Preliminaries on mathematical finance and numerical methods for stochastic differential equations

In this section, we review the basics of mathematical finance (including stochastic volatility, exotic options and no-arbitrage pricing) as well as numerical methods for stochastic differential equations (SDEs). Readers who are familiar with these contents can skip to the next section.

2.1 Mathematical finance

2.1.1 Stochastic volatility

The seminal Black-Scholes model [14, 15] assumes that the price S_t of the asset follows a geometric Brownian motion with constant drift and volatility:

$$dS_t = rS_t dt + \sigma S_t dW_t, \quad (1)$$

under the risk-neutral measure, where W_t is a Brownian motion under this measure¹, r is the risk-free interest rate, and σ is the volatility of the process. Then by Ito's lemma, one can prove that the asset price at any time t is given by

$$S_t = S_0 \exp \left(\left(r - \frac{\sigma^2}{2} \right) t + \sigma W_t \right) \quad (2)$$

and obeys a log-normal distribution [12]. This knowledge enables us to derive analytical formulas for the prices of certain options under the Black-Scholes model. For example, the value of an European call option with current underlying price S , strike price K and time to expiration τ is given by:

$$C(S, \tau) = SN(d_+) - Ke^{-r\tau}N(d_-), \quad (3)$$

where

$$d_{\pm} = \frac{1}{\sigma\sqrt{\tau}} \left[\ln \left(\frac{S}{K} \right) + \tau \left(r \pm \frac{\sigma^2}{2} \right) \right], \quad (4)$$

and $N(y) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^y e^{-z^2/2} dz$ is the cumulative distribution function (cdf) of a standard normal distribution. There also exist analytical formulas for the prices of barrier, digital and geometric Asian options under the Black-Scholes model. However, the pricing of other exotic options, e.g., arithmetic Asian options, and American options is more difficult and often requires numerical methods, e.g., Monte Carlo simulations, finite difference methods, and the binomial tree method [10, 11].

Although the Black-Scholes model is easy to use, it relies on the unrealistic assumption that the volatility remains a constant throughout the process. In reality, the volatility of asset prices not only fluctuates over time but also exhibits random behaviors. To better capture this phenomenon, various stochastic volatility models were proposed, and the **Heston model** [16] is one of the most popular among them. In this model, the instantaneous variance of the asset price follows a Cox–Ingersoll–Ross (CIR) process and exhibits

¹We can use Girsanov's theorem to convert a process $dS_t = \mu S_t dt + \sigma S_t dW_t$ under the real-world measure to an equivalent process $dS_t = r S_t dt + \sigma S_t d\tilde{W}_t$ under the risk-neutral measure. In this paper, we always work under the risk-neutral measure and hence drop the tilde notation for better readability.

mean reversion towards a long-term value. Formally, under the risk-neutral measure, the dynamics of the asset price S_t is governed by a system of stochastic differential equations:

$$dS_t = rS_t dt + \sqrt{\nu_t} S_t dW_t^1, \quad (5)$$

$$d\nu_t = \kappa(\theta - \nu_t) dt + \xi \sqrt{\nu_t} dW_t^2. \quad (6)$$

where r is the interest rate, θ is the long-term variance, κ is the rate of mean reversion, and ξ is the volatility of the price volatility $\sqrt{\nu_t}$. It is known that if the parameters obey the Feller condition

$$2\kappa\theta > \xi^2, \quad (7)$$

then the process ν_t is strictly positive. Henceforth, we will assume that this condition holds. Moreover, W_t^1 and W_t^2 are two Brownian motions with correlation coefficient $\rho \in [-1, 1]$, i.e., $\mathbb{E}[W_t^1 W_t^2] = \rho t$ (or $dW_t^1 dW_t^2 = \rho dt$)². Figure 1 illustrates an example of the evolution of asset price and its variance under the Heston model.

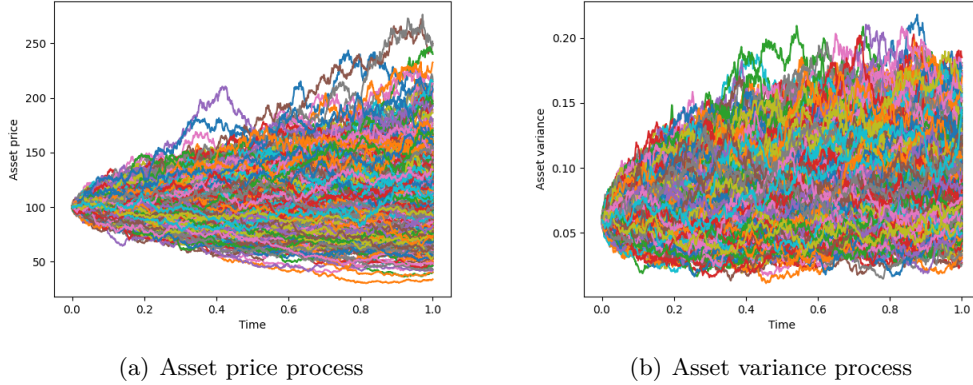


Figure 1: The evolution of asset price and its variance under the Heston model, in the case $S_0 = 100$, $\nu_0 = 0.06$, $r = 0.05$, $\rho = -0.1$, $\kappa = 2$, $\theta = 0.09$ and $\xi = 0.2$, from time 0 to time 1. Here this stochastic process is simulated by strong Euler method with 1024 time steps, and 1000 random paths are shown in the plots.

Let $R_t = \ln(S_t/S_0)$ be the log return at time t . We are often more interested in R_t than S_t itself. Using Ito's lemma, we can show that R_t follows the stochastic process:

$$dR_t = \left(r - \frac{\nu_t}{2}\right) dt + \sqrt{\nu_t} dW_t^1. \quad (8)$$

For technical reasons, it is more convenient to work with two independent Brownian motions instead of two correlated ones. Fortunately, we can find another Brownian motion V_t such that W_t^2 and V_t are independent and W_t^1 can be treated as a mixture of W_t^2 and V_t :

$$dW_t^1 = \rho dW_t^2 + \sqrt{1 - \rho^2} dV_t. \quad (9)$$

²Empirical evidence suggests that for stocks, ρ is often negative, indicating a tendency for the asset price and its volatility to be negatively correlated. This reflects the common observation that volatility tends to increase as the asset price decreases, and vice versa, a phenomenon often referred to as the *leverage effect* in financial markets.

As a consequence, we can re-write dR_t as

$$dR_t = \left(r - \frac{\nu_t}{2}\right) dt + \sqrt{\nu_t} \left[\rho dW_t^2 + \sqrt{1 - \rho^2} dV_t\right]. \quad (10)$$

In general, we do not have analytical expressions for the joint distribution of R_t and ν_t (or even just the distribution of R_t), although we know that the distribution of ν_t is related to a non-central chi-squared distribution and approaches a Gamma distribution as time becomes large. As a consequence, we need to resort to numerical methods, especially Monte Carlo simulations, to price many options under the Heston model ³.

2.1.2 Exotic options

An option is called *vanilla* or *path-independent* if its payoff depends only on the price of the underlying asset at the expiration date; otherwise, it is called *exotic* or *path-dependent*, as its payoff depends on the path of the asset price between now and the expiration date. Moreover, an option can be American-style or European-style. The former allows holders to exercise their rights at any time before or on the expiration date, while the latter only allows holders to do so on the expiration date. We will consider European-style exotic options in this work.

Specifically, we will focus on the pricing of two exotic options - (arithmetic) Asian option and barrier option - under the Heston model. These options play a major role in quantitative finance, not only as intensively traded contracts on their own, but also as the building blocks of a large variety of structured products. Meanwhile, there are no known analytic formulas for their prices under the Heston model. Classically, one often needs to utilize Monte Carlo simulation to estimate their prices. Quantum computing offers the potential to solve these tasks faster.

Asian options are a type of exotic options where the payoff depends on the average price of the underlying asset over a certain period of time instead of its price at expiration.

Formally, let $A_T = \frac{1}{T} \int_0^T S_t dt$ be the average price between time 0 and T . Then the payoff of an Asian call option with strike K and expiration time T is given by

$$f(S_{0:T}) = (A_T - K)^+, \quad (11)$$

where we use the notation $S_{0:T}$ to denote the path of asset price from time 0 to T , and $x^+ = \max(x, 0)$ is conventional. Similarly, the payoff of an Asian put option with strike K and expiration time T is given by

$$f(S_{0:T}) = (K - A_T)^+. \quad (12)$$

Barrier options are a type of exotic option in which payout depends on whether or not the underlying asset reaches or exceeds a predetermined price, i.e., a barrier. A barrier option can be *knock-out* or *knock-in*. While the former expires worthless if the underlying asset exceeds the barrier, the opposite is true for the latter, i.e. it has nonzero value only if the underlying asset reaches the same level.

Formally, let $M_T = \max_{0 \leq t \leq T} S_t$ and $N_T = \min_{0 \leq t \leq T} S_t$ be the maximum and minimum prices of the asset between time 0 and T , respectively. Then the payoff of an up-and-in, up-and-out, down-and-in, or down-and-out call option with strike K , barrier B and expiration

³[16] gives a simple way to price European vanilla options under the Heston model. It is, to our knowledge, unknown whether the method can be generalized to price more sophisticated options.

time T is given by

$$f(S_{0:T}) = 1_{M_T \geq B} \cdot (S_T - K)^+, \quad (13)$$

$$f(S_{0:T}) = 1_{M_T \leq B} \cdot (S_T - K)^+, \quad (14)$$

$$f(S_{0:T}) = 1_{N_T \leq B} \cdot (S_T - K)^+, \quad (15)$$

$$f(S_{0:T}) = 1_{N_T \geq B} \cdot (S_T - K)^+, \quad (16)$$

respectively. The payoff of a knock-in or knock-out put option can be defined similarly. We simply replace $(S_T - K)^+$ with $(K - S_T)^+$ in the above equations.

2.1.3 No-arbitrage pricing

Financial derivatives are priced based on the principle that no trader should make risk-free profit by buying the derivative and selling a replicating portfolio or vice versa. That is, the market should be free of arbitrage. Thus, the price of the derivative should be set at the same level as the value of the replicating portfolio. It can be shown that the value of the replicating portfolio equals the expected discounted payoff of the derivative under the risk-neutral measure.

Formally, assuming that the risk-free interest rate is a constant r , the no-arbitrage price of a derivative with payoff function f and expiration time T is given by

$$C(x, v) = e^{-rT} \mathbb{E}[f(S_{0:T}) | S_0 = x, \nu_0 = v], \quad (17)$$

where $\mathbb{E}[\cdot]$ is the expectation under the risk-neutral measure, $S_0 = x$ and $\nu_0 = v$ are the spot price and variance at time 0.

There are normally no closed-form expressions for the value $C(x, v)$ of the derivative, and one needs to compute it numerically. There are two common ways to achieve so [12, 10]: (1) use Monte Carlo simulation to generate paths of the underlying asset under the risk-neutral measure and use these paths to estimate the risk-neutral expected discounted payoff; or (2) reduce the problem to a partial differential equation and solve this PDE by numerical techniques, e.g., finite difference. Here we will follow the first approach and investigate how much advantage we can gain by quantizing it. We leave it as future work to explore the quantum version of the second approach.

2.2 Numerical methods for SDEs

In this subsection, we briefly review the common numerical techniques for solving stochastic differential equations. For more details, please refer to [12].

Consider a system of SDEs:

$$dX_t^k = a^k(X_t)dt + \sum_{j=1}^m b^{k,j}(X_t)dW_t^j, \quad (18)$$

for $k = 1, 2, \dots, d$, where each a^k and $b^{k,j}$ are functions from $\mathbb{R}^d$ to $\mathbb{R}$, and $W_t^1, W_t^2, \dots, W_t^m$ are independent Brownian motions. *Note that here and henceforth the superscripts k and j are not exponents, but labels for the dimensions of the system and the Brownian motions, respectively.*

This system of SDEs describes a d -dimensional Ito process driven by m Brownian motions:

$$X_t = X_0 + \int_0^t a(X_s)ds + \sum_{j=1}^m \int_0^t b^j(X_s)dW_s^j, \quad (19)$$

where $X_t = (X_t^1, X_t^2, \dots, X_t^d)$, $a = (a^1, a^2, \dots, a^d)$, and $b^j = (b^{1,j}, b^{2,j}, \dots, b^{d,j})$ for $j = 1, 2, \dots, m$. Note that the first integral in Eq. (19) is a Riemann integral, while the others are Ito integrals. Our goal is to approximate the probability distribution of X_t for given initial condition X_0 ⁴.

2.2.1 Strong and weak convergence criteria

To simulate the Ito process Eq. (19), we partition the time interval $[0, T]$ into N subintervals $0 = \tau_0 < \tau_1 < \dots < \tau_N = T$ and iteratively construct random variables $Y_0, Y_1, \dots, Y_N$ such that Y_n is an approximation of X_{τ_n} for $n = 0, 1, \dots, N$. Here we only consider the simplest equidistant case, i.e., $\tau_n = nh$, where $h = T/N$, for each n .

For example, the Euler-Maruyama scheme recursively defines the Y_n 's as follows:

$$Y_{n+1} = Y_n + a(Y_n)h + \sum_{j=1}^m b^j(Y_n)\Delta W_n^j, \quad (20)$$

for $n = 0, 1, \dots, N-1$ with initial value $Y_0 = X_0$ and $\Delta W_n^j = W_{\tau_{n+1}}^j - W_{\tau_n}^j$. Note that the ΔW_n^j 's are independent and identically distributed (i.i.d.) Gaussian random variables with expected value zero and variance h .

In the stochastic setting, a time discrete approximation Y_n of an Ito process X_t can converge to the true process in two different senses, one strong and one weak.

Strong convergence. We say that Y_N converges to X_T in the strong sense with order $\gamma \in (0, \infty)$ if there exists a finite constant K such that

$$\mathbb{E}[\|X_T - Y_N\|] \leq Kh^\gamma. \quad (21)$$

In this case, the trajectories, i.e., the sample paths, of the approximation are close to those of the Ito process. This implies that the probability distributions of $f(X_T)$ and $f(Y_N)$ are also close as long as the function f satisfies some mild conditions. In fact, if f is Lipschitz continuous, i.e., there exists a constant $L > 0$ such that $|f(x) - f(y)| \leq L\|x - y\|$, for all $x, y \in \mathbb{R}^d$, then the strong convergence of Y_N to X_T immediately implies the convergence of $f(Y_N)$ to $f(X_T)$:

$$|\mathbb{E}[f(X_T) - f(Y_N)]| \leq \mathbb{E}[|f(X_T) - f(Y_N)|] \leq L\mathbb{E}[\|X_T - Y_N\|] \leq LKh^\gamma. \quad (22)$$

Weak convergence. While strong methods give faithful pathwise approximations of Ito processes, they are often expensive to implement. In many practical situations, one is only interested in the expectation of some function f of the final value X_T of the Ito process, and it suffices to just have a good approximation of the probability distribution of $f(X_T)$. This can be done more efficiently by weak methods.

We say that Y_N converges to X_T in the weak sense with order $\beta \in (0, \infty)$ if for every $g \in \mathcal{C}_P^{2(\beta+1)}$, there exists a finite constant K such that

$$|\mathbb{E}[g(X_T)] - \mathbb{E}[g(Y_N)]| \leq Kh^\beta. \quad (23)$$

Here $\mathcal{C}_P^l$ is the set of l -times continuously differentiable functions from $\mathbb{R}^d$ to $\mathbb{R}$ whose partial derivatives up to order l have polynomial growth. In particular, all polynomials belong to this space for every l .

Note that as $h \rightarrow 0$, Y_N and X_T will have similar moment properties, which implies that the distribution of Y_N will converge to the distribution of X_T . However, it is difficult to bound the rate of this convergence in general.

⁴In general, X_0 can be random. But in this work, it is sufficient to assume that X_0 is deterministic.

2.2.2 Strong approximations of Ito processes

Next, we introduce some well-known first- and second-order strong and weak methods for SDEs. These methods rely on the following differential operators:

$$L^0 = \frac{\partial}{\partial t} + \sum_{k=1}^d a^k \frac{\partial}{\partial x^k} + \frac{1}{2} \sum_{k,l=1}^d \sum_{j=1}^m b^{k,j} b^{l,j} \frac{\partial^2}{\partial x^k \partial x^l}, \quad (24)$$

and

$$L^j = \sum_{k=1}^d b^{k,j} \frac{\partial}{\partial x^k}, \quad \forall 1 \leq j \leq m. \quad (25)$$

The Euler-Maruyama scheme is also known as the **strong Euler scheme** and has order $\gamma = 0.5$. Here we repeat its update rules:

$$Y_{n+1}^k = Y_n^k + a^k h + \sum_{j=1}^m b^{k,j} \Delta W_n^j, \quad (26)$$

for $k = 1, 2, \dots, d$ and $n = 0, 1, \dots, N-1$, where $\Delta W_n^j \sim \mathcal{N}(0, h)$ for each j , and the ΔW_n^j 's are independent from each other.

The **Milstein scheme** has strong order $\gamma = 1.0$ and has the following update rules:

$$Y_{n+1}^k = Y_n^k + a^k h + \sum_{j=1}^m b^{k,j} \Delta W_n^j + \sum_{j_1, j_2=1}^m L^{j_1} b^{k, j_2} I_n^{j_1, j_2}, \quad (27)$$

for $k = 1, 2, \dots, d$ and $n = 0, 1, \dots, N-1$, where

$$I_n^{j_1, j_2} = \int_{\tau_n}^{\tau_{n+1}} \int_{\tau_n}^{s_2} dW_{s_1}^{j_1} dW_{s_2}^{j_2} \quad (28)$$

for $j_1, j_2 \in \{1, 2, \dots, m\}$. When $j_1 = j_2 = j$, we have

$$I_n^{j,j} = \frac{1}{2} [(\Delta W_n^j)^2 - h]. \quad (29)$$

But when $j_1 \neq j_2$, the double Ito integral $I_n^{j_1, j_2}$ cannot be easily expressed in terms of the Brownian increments $\Delta W_n^{j_1}$ and $\Delta W_n^{j_2}$. This term is avoided only when $m = 1$, in which case the update rules become:

$$Y_{n+1}^k = Y_n^k + a^k h + b^k \Delta W_n + \frac{1}{2} \left(\sum_{l=1}^d b^l \frac{\partial b^k}{\partial x^l} \right) [(\Delta W_n)^2 - h]. \quad (30)$$

In this work, we need to simulate the Heston model which is a 2-dimensional SDE system, and it will be quite expensive to implement the Milstein scheme for this system. So we will not develop a quantum algorithm based on this scheme here. But we remark that one could utilize this method to simulate other stochastic volatility models such as the constant elasticity of variance (CEV) model.

2.2.3 Weak approximations of Ito processes

In weak methods, we have more degrees of freedom than with strong methods and can replace the increments ΔW_n^j 's and multiple Ito integrals (if any) in strong methods by more convenient random variables with similar moment properties.

Specifically, the (simplified) **weak Euler scheme** has order $\beta = 1.0$ and has the following update rules:

$$Y_{n+1}^k = Y_n^k + a^k h + \sum_{j=1}^m b^{k,j} \Delta \hat{W}_n^j, \quad (31)$$

for $k = 1, 2, \dots, d$ and $n = 0, 1, \dots, N - 1$, where

$$\mathbb{P}(\Delta \hat{W}_n^j = \pm \sqrt{h}) = \frac{1}{2} \quad (32)$$

for each j , and the $\Delta \hat{W}_n^j$'s are independent from each other. Note that $\Delta \hat{W}_n^j$ has mean 0 and variance h which are the same as those of ΔW_n^j in strong Euler scheme. More importantly, it is much easier to generate the discrete random variables $\Delta \hat{W}_n^j$'s than the continuous random variables ΔW_n^j 's both classically and quantumly.

The (simplified) **order 2.0 weak Taylor scheme** [12] has more complicated update rules:

$$\begin{aligned} Y_{n+1}^k = & Y_n^k + a^k h + \frac{1}{2} L^0 a^k h^2 \\ & + \sum_{j=1}^m \left\{ b^{k,j} + \frac{1}{2} h (L^0 b^{k,j} + L^j a^k) \right\} \Delta \hat{W}_n^j \\ & + \frac{1}{2} \sum_{j_1, j_2=1}^m L^{j_1} b^{k, j_2} (\Delta \hat{W}_n^{j_1} \Delta \hat{W}_n^{j_2} + V_n^{j_1, j_2}), \end{aligned} \quad (33)$$

for $k = 1, 2, \dots, d$ and $n = 0, 1, \dots, N - 1$, where

$$\mathbb{P}(\Delta \hat{W}_n^j = \pm \sqrt{3h}) = \frac{1}{6}, \quad \mathbb{P}(\Delta \hat{W}_n^j = 0) = \frac{2}{3}, \quad (34)$$

and

$$\mathbb{P}(V_n^{j_1, j_2} = \pm h) = \frac{1}{2}, \quad \text{for } 1 \leq j_2 < j_1, \quad (35)$$

$$V_n^{j_1, j_1} = -h, \quad (36)$$

$$V_n^{j_1, j_2} = -V_n^{j_2, j_1}, \quad \text{for } j_1 < j_2 \leq m, \quad (37)$$

for $j_1 = 1, 2, \dots, m$. The $\Delta \hat{W}_n^j$'s and $V_n^{j_1, j_2}$'s for $1 \leq j_1 < j_2 \leq m$ are independent from each other. Note again that it is fairly easy to generate the discrete random variables $\Delta \hat{W}_n^j$'s and $V_n^{j_1, j_2}$'s on both classical and quantum computers.

3 Numerical methods for option pricing under the Heston model

In this section, we develop three methods for simulating the Heston model which are based on the strong Euler, weak Euler and order 2.0 weak Taylor schemes for SDEs, respectively. Then we apply them to price Asian and barrier options under the same model, and empirically evaluate the performance of these methods in these tasks. Our simulation results suggest that the weak schemes perform as well as (or even better than)

strong Euler scheme. Furthermore, we do not see any advantage of order 2.0 weak Taylor scheme over the other schemes in our experiments ⁵. It turns out that the simplest weak Euler scheme is the most cost-effective choice among the three.

3.1 Numerical simulation of the Heston model

To employ the numerical methods of Section 2.2 to simulate the Heston model, we need to first convert the model into the standard form Eq. (18). Let $X_t = (X_t^1, X_t^2) = (R_t, \nu_t)$ and $B_t = (B_t^1, B_t^2) = (W_t^2, V_t)$. Then we can rewrite the Heston model as the following 2-dimensional SDE system:

$$dX_t^1 = a^1(X_t)dt + b^{1,1}(X_t)dB_t^1 + b^{1,2}(X_t)dB_t^2, \quad (38)$$

$$dX_t^2 = a^2(X_t)dt + b^{2,1}(X_t)dB_t^1 + b^{2,2}(X_t)dB_t^2, \quad (39)$$

where

$$a^1(X_t) = r - \frac{1}{2}\nu_t, \quad b^{1,1}(X_t) = \rho\sqrt{\nu_t}, \quad b^{1,2}(X_t) = \sqrt{1 - \rho^2}\sqrt{\nu_t}, \quad (40)$$

$$a^2(X_t) = \kappa(\theta - \nu_t), \quad b^{2,1}(X_t) = \xi\sqrt{\nu_t}, \quad b^{2,2}(X_t) = 0. \quad (41)$$

It follows that

$$\frac{\partial a^1}{\partial \nu_t} = -\frac{1}{2}, \quad (42)$$

$$\frac{\partial b^{1,1}}{\partial \nu_t} = \frac{1}{2}\rho\nu_t^{-1/2}, \quad \frac{\partial^2 b^{1,1}}{\partial \nu_t^2} = -\frac{1}{4}\rho\nu_t^{-3/2}, \quad (43)$$

$$\frac{\partial b^{1,2}}{\partial \nu_t} = \frac{1}{2}\sqrt{1 - \rho^2}\nu_t^{-1/2}, \quad \frac{\partial^2 b^{1,2}}{\partial \nu_t^2} = -\frac{1}{4}\sqrt{1 - \rho^2}\nu_t^{-3/2}, \quad (44)$$

$$\frac{\partial a^2}{\partial \nu_t} = -\kappa, \quad (45)$$

$$\frac{\partial b^{2,1}}{\partial \nu_t} = \frac{1}{2}\xi\nu_t^{-1/2}, \quad \frac{\partial^2 b^{2,1}}{\partial \nu_t^2} = -\frac{1}{4}\xi\nu_t^{-3/2}. \quad (46)$$

All the other up-to-second-order partial derivatives of a^k or $b^{k,j}$ with respect to R_t or ν_t are zero.

Next, we define the differential operators L^0 , L^1 and L^2 as in Eqs. (24) and (25) (here we have two variables $x^1 = R_t$ and $x^2 = \nu_t$). Using the facts:

1. Applying $\frac{\partial}{\partial x^1}$ to a^k or $b^{k,j}$ gives 0;
2. Applying $\frac{\partial^2}{\partial x^i \partial x^j}$ to a^k or $b^{k,l}$ yields 0 unless $i = j = 2$;
3. $b^{2,2} = 0$,

we can simplify L^0 , L^1 and L^2 into:

$$L^0 = \frac{\partial}{\partial t} + a^2 \frac{\partial}{\partial x^2} + \frac{1}{2}(b^{2,1})^2 \frac{\partial^2}{\partial (x^2)^2}, \quad (47)$$

$$L^1 = b^{2,1} \frac{\partial}{\partial x^2}, \quad (48)$$

$$L^2 = 0. \quad (49)$$

⁵It is possible, however, that high-order methods do have advantages over first-order ones when one wants to estimate the option's value within very high accuracy.

Now we apply three numerical methods to the Heston model. In all of these methods, we start with $Y_0^1 = 0$ and $Y_0^2 = \nu_0$. Then they will be updated differently:

1. Strong Euler method: The update rules are

$$Y_{n+1}^k = Y_n^k + a^k h + \sum_{j=1}^2 b^{k,j} \Delta W_n^j, \quad (50)$$

for $k = 1, 2$ and $n = 0, 1, \dots, N-1$, where $\Delta W_n^j \sim \mathcal{N}(0, h)$, and the ΔW_n^j 's are independent from each other;

2. Weak Euler method: The update rules are

$$Y_{n+1}^k = Y_n^k + a^k h + \sum_{j=1}^2 b^{k,j} \Delta \hat{W}_n^j, \quad (51)$$

for $k = 1, 2$ and $n = 0, 1, \dots, N-1$, where

$$\mathbb{P}(\Delta \hat{W}_n^j = \pm \sqrt{h}) = \frac{1}{2}, \quad (52)$$

and the $\Delta \hat{W}_n^j$'s are independent from each other;

3. Order 2.0 weak Taylor method: The update rules are

$$\begin{aligned} Y_{n+1}^k = & Y_n^k + a^k h + \frac{1}{2} L^0 a^k h^2 + \sum_{j=1}^2 \left[b^{k,j} + \frac{1}{2} h (L^0 b^{k,j} + L^j a^k) \right] \Delta \hat{W}_n^j \\ & + \frac{1}{2} \sum_{j_1, j_2=1}^2 L^{j_1} b^{k, j_2} (\Delta \hat{W}_n^{j_1} \Delta \hat{W}_n^{j_2} + V_n^{j_1, j_2}) \end{aligned} \quad (53)$$

$$\begin{aligned} = & Y_n^k + a^k h + \sum_{j=1}^2 b^{k,j} \Delta \hat{W}_n^j + \frac{1}{2} L^0 a^k h^2 + \frac{1}{2} \sum_{j=1}^2 h (L^0 b^{k,j} + L^j a^k) \Delta \hat{W}_n^j \\ & + \frac{1}{2} \sum_{j_2=1}^2 L^1 b^{k, j_2} (\Delta \hat{W}_n^1 \Delta \hat{W}_n^{j_2} + V_n^{1, j_2}), \end{aligned} \quad (54)$$

for $k = 1, 2$ and $n = 0, 1, \dots, N-1$, where the second step follows from $L^2 = 0$, and

$$\mathbb{P}(\Delta \hat{W}_n^j = \pm \sqrt{3h}) = \frac{1}{6}, \quad \mathbb{P}(\Delta \hat{W}_n^j = 0) = \frac{1}{3}, \quad (55)$$

and $V_n^{1,1} = V_n^{2,2} = -h$, $V_n^{1,2} = -V_n^{2,1}$, $\mathbb{P}(V_n^{1,2} = \pm h) = 1/2$. The $\Delta \hat{W}_n^j$'s and $V_n^{1,2}$'s are independent from each other. We can further simplify the update rules into:

$$\begin{aligned} Y_{n+1}^1 = & Y_n^1 + a^1 h + b^{1,1} \Delta \hat{W}_n^1 + b^{1,2} \Delta \hat{W}_n^2 + \frac{1}{2} L^0 a^1 h^2 \\ & + \frac{1}{2} h [(L^0 b^{1,1} + L^1 a^1) \Delta \hat{W}_n^1 + L^0 b^{1,2} \Delta \hat{W}_n^2] \\ & + \frac{1}{2} L^1 b^{1,1} [(\Delta \hat{W}_n^1)^2 - h] + \frac{1}{2} L^1 b^{1,2} (\Delta \hat{W}_n^1 \Delta \hat{W}_n^2 + V_n^{1,2}), \end{aligned} \quad (56)$$

$$\begin{aligned} Y_{n+1}^2 = & Y_n^2 + a^2 h + b^{2,1} \Delta \hat{W}_n^1 + \frac{1}{2} L^0 a^2 h^2 \\ & + \frac{1}{2} h (L^0 b^{2,1} + L^1 a^2) \Delta \hat{W}_n^1 + \frac{1}{2} L^1 b^{2,1} [(\Delta \hat{W}_n^1)^2 - h]. \end{aligned} \quad (57)$$

Given an option with payoff function f , we can use each of the above methods to generate a random path $\vec{Y} = (Y_0, Y_1, Y_2, \dots, Y_N)$, and translate them into the price path $\vec{S} = (S_0, S_h, S_{2h}, \dots, S_T)$, and then compute the payoff $f(\vec{S})$ of this path. By repeating this procedure sufficiently many times and computing the mean of the collected samples, we can estimate the value of this option under the Heston model within desired accuracy.

3.2 Empirical results

Next, we apply the methods in Section 3.1 to several instances of option pricing under the Heston model to test their performance in practice. Specifically, we consider four settings of the Heston model which are listed in Table 1 and estimate the values of an Asian option and a barrier option under each setting. The specifications of these options are given in Table 2.

Setting	r	ρ	κ	θ	ξ	S_0	ν_0
No. 1	0.03	-0.1	2	0.12	0.3	100	0.1
No. 2	0.03	0	2	0.03	0.2	100	0.03
No. 3	0.05	-0.1	2	0.09	0.2	100	0.06
No. 4	0.05	-0.1	2	0.04	0.2	100	0.05

Table 1: The settings of the Heston model studied in Section 3.2. Each setting includes the model parameters r , ρ , κ , θ , ξ and initial condition S_0 and ν_0 .

Instance	Setting	Option type	K	B	T
No. 1	No. 1	Asian call	90	N/A	1.0
No. 2	No. 1	Down-and-out put	110	70	1.0
No. 3	No. 2	Asian put	110	N/A	1.0
No. 4	No. 2	Up-and-out call	90	130	1.0
No. 5	No. 3	Asian call	90	N/A	1.0
No. 6	No. 3	Down-and-in put	110	80	1.0
No. 7	No. 4	Asian put	110	N/A	1.0
No. 8	No. 4	Up-and-in call	90	120	1.0

Table 2: The problem instances studied in Section 3.2. The specification of each instance includes the setting of the Heston model and the type, strike K , barrier B (if needed) and expiration time T of the option.

In each experiment, we utilize the strong Euler method, weak Euler method and weak Taylor method (order 2.0) to generate 5 million random paths, and compute the mean and standard deviation of the payoffs of the option for the collected paths in each case. We vary the number of time steps N in powers of two starting from 2, 4, 8, ... up to 1024, and observe how fast the estimated value converge to the true value as N grows large for each method ⁶.

Figures 2, 3, 4, 5, 6, 7, 8 and 9 illustrate the results of our experiments. In each plot, there are three curves with error bands for the strong Euler method, weak Euler

⁶We use the result of the strong Euler method with $N = 1024$ time steps as the best estimate of the true value, as this method is known for its robust performance in producing pathwise accurate simulations in a variety of stochastic models and has been widely employed as a benchmark in the numerical solution of SDEs [12, 25, 26, 27, 28, 29, 30]. Although this estimate contains a bias due to time discretization, the bias is small enough to be negligible for our purposes.

method and weak Taylor method (order 2.0) respectively. The upper and lower bands are 3 standard deviations $\pm$ from the mean of the 5 million samples. Furthermore, there is also a vertical line with error bands which indicates our best estimate of the true value which is obtained from the $N = 1024$ strong Euler method. The upper and lower bands are again 3 standard deviations $\pm$ from the mean of the 5 million samples in that case.

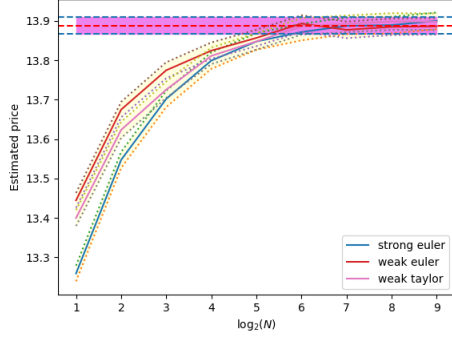


Figure 2: The estimated value of an Asian call with strike $K = 90$ and expiration time $T = 1$, in the case $S_0 = 100$, $\nu_0 = 0.1$, $r = 0.03$, $\rho = -0.1$, $\kappa = 2$, $\theta = 0.12$, $\xi = 0.3$, in different methods with varying number of time steps.

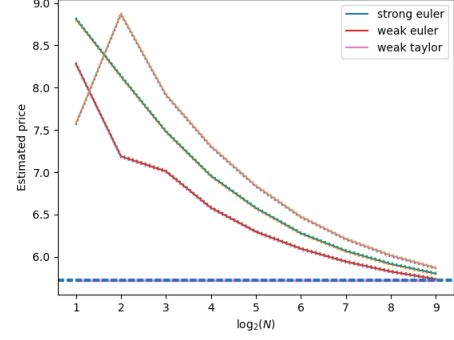


Figure 3: The estimated value of a down-and-out put with strike $K = 110$, barrier $B = 70$ and expiration time $T = 1$, in the case $S_0 = 100$, $\nu_0 = 0.1$, $r = 0.03$, $\rho = -0.1$, $\kappa = 2$, $\theta = 0.12$, $\xi = 0.3$, in different methods with varying number of time steps.

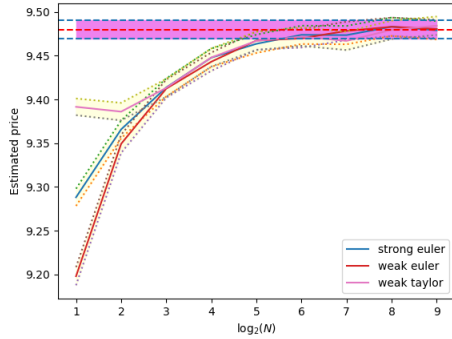


Figure 4: The estimated value of an Asian put with strike $K = 110$ and expiration time $T = 1$, in the case $S_0 = 100$, $\nu_0 = 0.03$, $r = 0.03$, $\rho = 0$, $\kappa = 2$, $\theta = 0.03$, $\xi = 0.2$, in different methods with varying number of time steps.

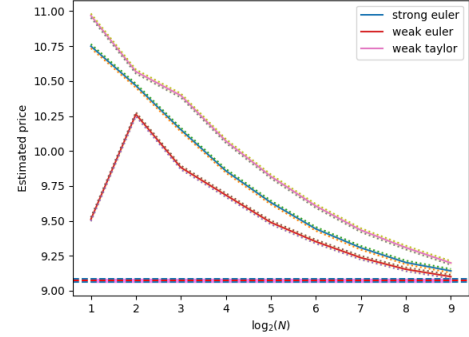


Figure 5: The estimated value of an up-and-out call with strike $K = 90$, barrier $B = 130$ and expiration time $T = 1$, in the case $S_0 = 100$, $\nu_0 = 0.03$, $r = 0.03$, $\rho = 0$, $\kappa = 2$, $\theta = 0.03$, $\xi = 0.2$, in different methods with varying number of time steps.

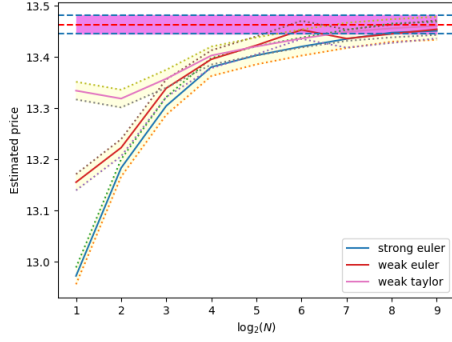


Figure 6: The estimated value of an Asian call with strike $K = 90$ and expiration time $T = 1$, in the case $S_0 = 100$, $\nu_0 = 0.06$, $r = 0.05$, $\rho = -0.1$, $\kappa = 2$, $\theta = 0.09$, $\xi = 0.2$, in different methods with varying number of time steps.

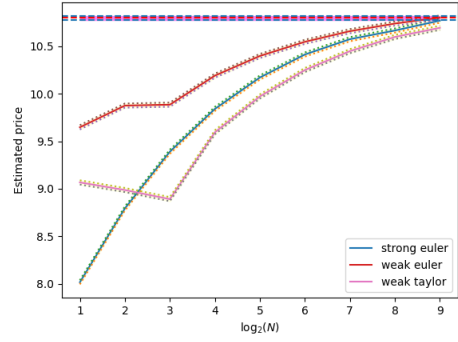


Figure 7: The estimated value of a down-and-in put with strike $K = 110$, barrier $B = 80$ and expiration time $T = 1$, in the case $S_0 = 100$, $\nu_0 = 0.06$, $r = 0.05$, $\rho = -0.1$, $\kappa = 2$, $\theta = 0.09$, $\xi = 0.2$, in different methods with varying number of time steps.

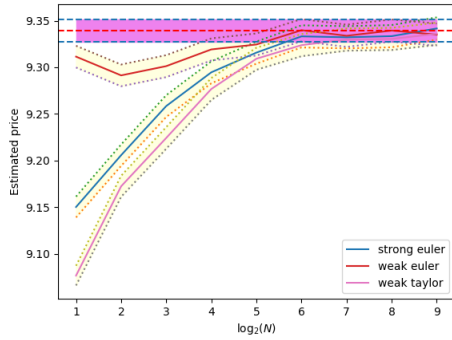


Figure 8: The estimated value of an Asian put with strike $K = 110$ and expiration time $T = 1$, in the case $S_0 = 100$, $\nu_0 = 0.05$, $r = 0.05$, $\rho = -0.1$, $\kappa = 2$, $\theta = 0.04$, $\xi = 0.2$, in different methods with varying number of time steps.

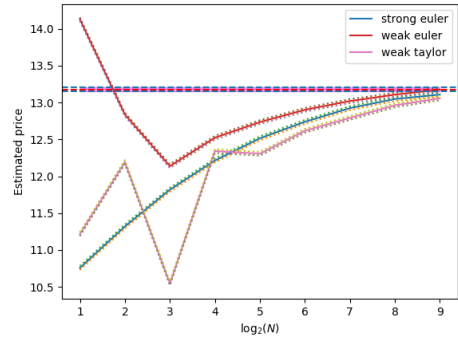


Figure 9: The estimated value of an up-and-in call with strike $K = 90$, barrier $B = 120$ and expiration time $T = 1$, in the case $S_0 = 100$, $\nu_0 = 0.05$, $r = 0.05$, $\rho = -0.1$, $\kappa = 2$, $\theta = 0.04$, $\xi = 0.2$, in different methods with varying number of time steps.

In all of the above experiments, the strong Euler, weak Euler, and order 2.0 weak Taylor methods exhibit similar performance. Note that for Asian and barrier options, their payoff functions depend on the path of the asset price between time 0 and time T . This differs from the traditional setting in numerical SDEs where the payoff function depends only on the asset price at time T . Thus, the strong Euler, weak Euler, and weak Taylor (order 2.0) methods might exhibit different convergence behaviors in this path-dependent setting, and there is currently no theoretical proof of their convergence orders in our specific tasks. Nevertheless, our simulation results indicate that the accuracy of these methods improves rapidly as N increases under typical market conditions.

Interestingly, despite its simplicity, the weak Euler method often performs best among the three methods (in the sense that its output converges to the ideal value faster than the others) in our experiments. This suggests that for pricing Asian and barrier options under the Heston model, it is not crucial to have a faithful pathwise approximation of the Ito process. Instead, it suffices to replace the real probability distribution of the price path

with one that has similar moment properties.

Moreover, we do not observe any significant advantage of the order 2.0 weak Taylor method over the strong and weak Euler methods. Note that we do not aim to estimate the option's value with extremely high accuracy, e.g., with an error of less than 0.001 in the above settings. It is possible that higher-order methods start to show their advantages over lower-order ones as we enter high-precision regimes, as predicted by existing literature, where higher-order Taylor methods tend to outperform others in high-precision scenarios or when very fine time discretization is required [12, 27, 28].

Our experiments also suggest that, in general, Asian options are easier to price than barrier options under the Heston model. This is reasonable because the payoffs of Asian options depend on the average price between time 0 and T , making them more robust against small deviations from the ideal paths in the simulations. In contrast, the payoffs of barrier options crucially depend on whether the price reaches a certain threshold during the process, and this event is more sensitive to the aforementioned deviations. Therefore, a more accurate simulation of the Ito process (which requires a larger N) is needed to achieve the same accuracy for barrier options as for Asian options.

4 Option pricing under the Heston model on quantum computers

In this section, we describe our quantum algorithms for pricing Asian and barrier options under the Heston model, and estimate how many resources, i.e., T-count, T-depth and number of qubits, they require to achieve certain accuracy in their outputs on certain input instances. These algorithms are based on combining quantum amplitude estimation [13, 21] and strong/weak Euler schemes for SDEs, respectively. Our resource estimation indicates that the quantum algorithms based on weak Euler scheme are far more efficient than the ones based on strong Euler scheme.

4.1 Basic strategy

Given a specification of the Heston model, including the parameters $r, \kappa, \theta, \xi, \rho$ and initial condition S_0, ν_0 , we first choose a numerical scheme for simulating its dynamics. Here we use either strong Euler or weak Euler methods (see Eqs. (50) and (51)). (As shown in Section 3.2, order 2.0 weak Taylor method does not seem to have an advantage over strong and weak Euler methods, despite being significantly more complicated. So we will not develop quantum algorithms for option pricing based on this numerical scheme.)

Let $Y_0^1 = 0$ and $Y_0^2 = \nu_0$, and let Y_n^k be defined recursively as Eq. (50) or Eq. (51) for $k = 1, 2$ and $n = 1, \dots, N$. Then Y_n^1 and Y_n^2 approximate the log return and asset variance at time nh in the ideal process, respectively, i.e.,

$$Y_n^1 \approx R_{nh} = \ln(S_{nh}/S_0), \quad Y_n^2 \approx \nu_{nh}, \quad (58)$$

provided that h is sufficiently small.

The input to our problem also includes the specification of an Asian or barrier option. Suppose it has payoff function f and expiration time T . Without loss of generality, we rescale f so that it takes values in $[0, 1]$. This facilitates the construction of quantum circuits for option pricing, because we want to encode the payoffs of the option for possible paths into quantum amplitudes and they cannot exceed 1.

Let $\vec{S} = (S_0, S_h, \dots, S_T)$ and $\vec{R} = (R_0, R_h, \dots, R_T)$ be the paths of asset price and log return in the ideal process, respectively (note that S_0 is fixed and $R_0 = 0$). Then the no-arbitrage price of the option is $e^{-rT} \mathbb{E}[f(\vec{S})]$. In fact, it is more convenient to work

with log returns than prices. So we re-define the payoff in terms of $\vec{R}$, i.e., there exists a function $\tilde{f}$ such that

$$\tilde{f}(\vec{R}) = f(\vec{S}). \quad (59)$$

Next, let $\vec{Y}^k = (Y_0^k, Y_1^k, \dots, Y_N^k)$ be the path of Y_n^k in the simulated process, for $k = 1, 2$. Then we get

$$e^{-rT} \mathbb{E} [\tilde{f}(\vec{R})] \approx e^{-rT} \mathbb{E} [\tilde{f}(\vec{Y}^1)]. \quad (60)$$

Our goal is to estimate $\mathbb{E} [\tilde{f}(\vec{Y}^1)]$ for the chosen numerical scheme. Note that the random path in the simulated process can be parameterized by a random variable ω drawn from a probability distribution $p(\omega)$ on a set Ω . For example, in strong Euler method, we define $\omega = (\Delta W_n^j / \sqrt{h} : 0 \leq n \leq N-1, 1 \leq j \leq 2)$, and p is the $2N$ -dimensional standard normal distribution on $\Omega = \mathbb{R}^{2N}$. Meanwhile, in weak Euler method, we define $\omega = (\Delta \hat{W}_n^j / \sqrt{h} : 0 \leq n \leq N-1, 1 \leq j \leq 2)$, and p is the uniform distribution on $\Omega = \{1, -1\}^{2N}$. Then each Y_n^k is a deterministic function of ω . Thus, our task is reduced to estimating

$$\mathbb{E} [\tilde{f}(\vec{Y}^1(\omega))] = \sum_{\omega \in \Omega} p(\omega) \tilde{f}(\vec{Y}^1(\omega)). \quad (61)$$

Here we abuse the notation for strong Euler method (which should have integration instead of summation) without causing confusion.

Classically, one often estimates $\mathbb{E} [\tilde{f}(\vec{Y}^1(\omega))]$ by Monte Carlo methods. That is, one simulates the stochastic process, and generates M random paths, and computes the mean of the payoffs of the option for these paths. This leads to an unbiased estimator of the target quantity whose standard deviation scales as $O(1/\sqrt{M})$, which means that we need to collect $O(1/\epsilon^2)$ samples to reach precision ϵ in our estimate.

Quantumly, we can utilize the technique of amplitude estimation [13] to achieve quadratic speedup in the estimation of the same quantity. Specifically, suppose $\mathcal{A}$ is a unitary operation such that

$$\mathcal{A} |0\rangle |0^m\rangle = \sqrt{a} |0\rangle |\psi_0\rangle + \sqrt{1-a} |1\rangle |\psi_1\rangle, \quad (62)$$

where $a = \mathbb{E} [\tilde{f}(\vec{Y}^1(\omega))]$ is the target quantity, and $|\psi_0\rangle$ and $|\psi_1\rangle$ are some normalized m -qubit states. Quantum amplitude estimation (QAE) can estimate a within additive error ϵ with probability $1 - O(1)$ by making $O(1/\epsilon)$ uses of the unitary operation

$$Q = \mathcal{A} R_0 \mathcal{A}^{-1} S_0, \quad (63)$$

where $R_0 = I_{m+1} - 2 |0^{m+1}\rangle \langle 0^{m+1}|$ ⁷ and $S_0 = I_{m+1} - 2 |0\rangle \langle 0| \otimes I_m$ are reflection operators. QAE consists of running quantum phase estimation (QPE) on the unitary operator Q with initial state $\mathcal{A} |0^{m+1}\rangle$ and inferring a from the measurement outcomes of this circuit.

A drawback of standard Quantum Amplitude Estimation (QAE) is that it requires performing a quantum Fourier transform on multiple ancilla qubits, which is expensive to implement. To overcome this limitation, several variants of QAE have been proposed [31, 32, 33, 21, 34, 35] that do not involve the use of ancilla qubits or quantum Fourier transform. In particular, the iterative quantum amplitude estimation (IQAE) of [21] has

⁷Following convention, we use I_n to denote the n -qubit identity operator.

shown excellent empirical performance. Therefore, we will utilize this algorithm for our task. This algorithm can estimate a within additive error ϵ with probability $1 - \delta$ by making

$$N_{\text{oracle}} \leq \frac{1.4}{\epsilon} \ln \left(\frac{2}{\delta} \log_2 \left(\frac{\pi}{4\epsilon} \right) \right) \quad (64)$$

uses of Q , for any $\epsilon > 0$ and $\delta \in (0, 1)$.

Now our task is reduced to building a quantum circuit $\mathcal{A}$ that satisfies the condition Eq. (62). This can be achieved by concatenating three unitary operations U_1 , U_2 and U_3 , where U_1 simulates the Heston model and creates a proper superposition of the log-return paths, U_2 calculates the payoffs of the option for these paths simultaneously, and U_3 encodes the payoffs into quantum amplitudes. Formally, we define

$$\mathcal{A} = U_3 U_2 U_1 \quad (65)$$

where

$$U_1 |00 \dots 0\rangle_A |00 \dots 0\rangle_F = \sum_{\omega \in \Omega} \sqrt{p(\omega)} |\vec{Y}^1(\omega)\rangle_A |\Phi(\omega)\rangle_F, \quad (66)$$

$$U_2 |\vec{Y}^1(\omega)\rangle_A |0\rangle_C |00 \dots 0\rangle_G = |\vec{Y}^1(\omega)\rangle_A |\tilde{f}(\vec{Y}^1(\omega))\rangle_C |\Psi(\omega)\rangle_G, \quad \forall \omega \in \Omega, \quad (67)$$

$$U_3 |z\rangle_C |0\rangle_D |00 \dots 0\rangle_H = |z\rangle_C [\sqrt{z}|0\rangle_D + \sqrt{1-z}|1\rangle_D] |\Xi(z)\rangle_H, \quad \forall z \in [0, 1]. \quad (68)$$

Here $|\Phi(\omega)\rangle$ and $|\Psi(\omega)\rangle$ are some normalized states depending on ω , and $|\Xi(z)\rangle$ is a normalized state depending on z . One can verify that

$$\|(I_{ACFGH} \otimes \langle 0|_D) \mathcal{A} |00 \dots 0\rangle_{ACDFGH}\|^2 = \mathbb{E} [\tilde{f}(\vec{Y}^1(\omega))]^2, \quad (69)$$

as desired.

In the next three subsections, we will construct the circuits for U_1 , U_2 and U_3 , respectively. Each circuit is composed of the basic modules introduced in Appendices A, B and C, including various fixed-point quantum arithmetic operations, a block-encoding U_{sin} of the sine function, a unitary operation U_{gauss} for preparing Gaussian states (for strong Euler method only), as well as Clifford gates. The costs of these modules are also given in these appendices. Here we assume Clifford + T gate set, and measure the cost of an operation in terms of the T-count, T-depth and number of ancilla qubits in the circuit for implementing this operation. Every real number will be represented by n qubits in the circuits (see Appendix A for more details). To avoid distraction from our main messages, we will ignore the errors in quantum arithmetic operations, U_{sin} and U_{gauss} for now, and postpone the error analysis to Section 4.6.

4.2 Implementing the operator U_1

To facilitate the implementation of U_1 , we re-write the update rules of strong Euler method as

$$Y_{j+1}^1 = Y_j^1 - Y_j^2 \cdot \frac{h}{2} + \alpha_j \cdot \sqrt{Y_j^2} \cdot \rho \sqrt{h} + \beta_j \cdot \sqrt{Y_j^2} \cdot \sqrt{h(1-\rho^2)} + rh, \quad (70)$$

$$Y_{j+1}^2 = Y_j^2 - Y_j^2 \cdot \kappa h + \alpha_j \cdot \sqrt{Y_j^2} \cdot \xi \sqrt{h} + \kappa \theta h, \quad (71)$$

where $\alpha_j, \beta_j \sim N(0, 1)$, and re-write the update rules of weak Euler method as

$$Y_{j+1}^1 = Y_j^1 - Y_j^2 \cdot \frac{h}{2} + (-1)^{\alpha_j} \cdot \sqrt{Y_j^2} \cdot \rho \sqrt{h} + (-1)^{\beta_j} \cdot \sqrt{Y_j^2} \cdot \sqrt{h(1 - \rho^2)} + rh, \quad (72)$$

$$Y_{j+1}^2 = Y_j^2 - Y_j^2 \cdot \kappa h + (-1)^{\alpha_j} \cdot \sqrt{Y_j^2} \cdot \xi \sqrt{h} + \kappa \theta h, \quad (73)$$

where $\alpha_j, \beta_j = 0$ or 1 with probability $1/2$. Note that the blue terms in the above equations are constants and can be pre-computed classically.

U_1 for strong Euler scheme. We implement U_1 as the product of $N + 2$ unitary operations:

$$U_1 = V_{N-1} V_{N-2} \dots V_1 V_0 Q_2 Q_1. \quad (74)$$

The goal of Q_1 and Q_2 is to prepare the initial state and load the necessary Gaussian distributions for α_j 's and β_j 's, respectively. Then $V_0, V_1, \dots, V_{N-1}$ performs the N iterations of updates in strong Euler scheme.

Specifically, Q_1 prepares the initial state $|Y_0^1\rangle_{A_1} |Y_0^2\rangle_{B_1}$:

$$Q_1 |0\rangle_{A_1} |0\rangle_{B_1} = |Y_0^1\rangle_{A_1} |Y_0^2\rangle_{B_1} = |0\rangle_{A_1} |\nu_0\rangle_{B_1} \quad (75)$$

This operation can be implemented with X gates only.

Meanwhile, Q_2 creates a quantum state that encodes the distribution of $(\vec{\alpha}, \vec{\beta})$, where $\vec{\alpha} = (\alpha_0, \alpha_1, \dots, \alpha_{N-1})$ and $\vec{\beta} = (\beta_0, \beta_1, \dots, \beta_{N-1})$. Ideally, each α_i and β_i should have standard normal distribution. But on a real computer, they can only take a finite number of possible values. Thus, we have to approximate the original continuous distribution with a discrete distribution. To this end, we introduce a unitary operation U_{gauss} such that

$$U_{\text{gauss}} |0\rangle \approx \frac{1}{Z} \sum_{i=0}^{M-1} e^{-\frac{x_i^2}{4}} |x_i\rangle, \quad (76)$$

where $x_i = (-1 + \frac{2i}{M})\eta$ for $0 \leq i \leq M-1$, $Z = \sum_{i=0}^{M-1} e^{-x_i^2/2}$, and M, η are appropriately chosen parameters. The implementation of U_{gauss} is given in Appendix C. Then we define $Q_2 = U_{\text{gauss}}^{\otimes 2N}$ and perform it on $2N$ n -qubit registers $E_0, E_1, \dots, E_{2N-2}, E_{2N-1}$, obtaining

$$Q_2 |00 \dots 0\rangle_{E_0 E_1 \dots E_{2N-2} E_{2N-1}} = \sum_{\vec{\alpha}, \vec{\beta}} \sqrt{\tilde{p}(\vec{\alpha}, \vec{\beta})} |\alpha_0\rangle_{E_0} |\beta_0\rangle_{E_1} \dots |\alpha_{N-1}\rangle_{E_{2N-2}} |\beta_{N-1}\rangle_{E_{2N-1}}, \quad (77)$$

where $\tilde{p}$ is a discrete approximation of the $2N$ -dimensional standard normal distribution.

Next, for $j = 0, 1, \dots, N-1$, we define V_j as a unitary operation satisfying:

$$\begin{aligned} V_j & |Y_j^1\rangle_{A_j} |Y_j^2\rangle_{B_j} |\alpha_j\rangle_{E_{2j}} |\beta_j\rangle_{E_{2j+1}} |0\rangle_{A_{j+1}} |0\rangle_{B_{j+1}} \\ &= |Y_j^1\rangle_{A_j} |Y_j^2\rangle_{B_j} |\alpha_j\rangle_{E_{2j}} |\beta_j\rangle_{E_{2j+1}} |Y_{j+1}^1\rangle_{A_{j+1}} |Y_{j+1}^2\rangle_{B_{j+1}}. \end{aligned} \quad (78)$$

That is, it computes Y_{j+1}^1 and Y_{j+1}^2 from Y_j^1, Y_j^2, α_j and β_j . We implement V_j by concatenating 8 unitary operations:

$$V_j = R_j^\dagger S_j^\dagger T_j^\dagger H_j T_j S_j R_j G_j. \quad (79)$$

These operations work as follows:

1. G_j copies Y_j^1 and Y_j^2 to registers A_{j+1} and B_{j+1} , respectively:

$$G_j |Y_j^1\rangle_{A_j} |Y_j^2\rangle_{B_j} |0\rangle_{A_{j+1}} |0\rangle_{B_{j+1}} = |Y_j^1\rangle_{A_j} |Y_j^2\rangle_{B_j} |Y_j^1\rangle_{A_{j+1}} |Y_j^2\rangle_{B_{j+1}} \quad (80)$$

This step requires only CNOT gates.

2. R_j computes the square root of Y_j^2 and saves it in an ancilla register J :

$$R_j |Y_j^2\rangle_{B_j} |0\rangle_J = |Y_j^2\rangle_{B_n} |\sqrt{Y_j^2}\rangle_J. \quad (81)$$

This step requires a square root operation.

3. Let $\vec{\Lambda}_j = (\alpha_j \sqrt{Y_j^2}, \beta_j \sqrt{Y_j^2})$. Then S_j computes these terms and saves them in an ancilla register L :

$$S_j |\sqrt{Y_j^2}\rangle_J |\alpha_j\rangle_{E_{2j}} |\beta_j\rangle_{E_{2j+1}} |\vec{0}\rangle_L = |\sqrt{Y_j^2}\rangle_J |\alpha_j\rangle_{E_{2j}} |\beta_j\rangle_{E_{2j+1}} |\vec{\Lambda}_j\rangle_L \quad (82)$$

This step requires 2 multiplication operations.

4. Let $\vec{\Gamma}_j = (-Y_j^2 \cdot \frac{h}{2}, \alpha_j \sqrt{Y_j^2} \cdot \rho \sqrt{h}, \beta_j \sqrt{Y_j^2} \cdot \sqrt{h(1-\rho^2)}, -Y_j^2 \cdot \kappa h, \alpha_j \sqrt{Y_j^2} \cdot \xi \sqrt{h})$. Then T_j computes these terms and saves them in an ancilla register M :

$$T_j |Y_j^2\rangle_{B_j} |\vec{\Lambda}_j\rangle_L |\vec{0}\rangle_M = |Y_j^2\rangle_{B_j} |\vec{\Lambda}_j\rangle_L |\vec{\Gamma}_j\rangle_M \quad (83)$$

This step requires 5 multiplication-with-a-constant operations.

5. H_j adds the terms in $\vec{\Gamma}_j$ and the two constant terms $rh, \kappa\theta h$ to the values in registers A_{j+1} and B_{j+1} , transforming them into Y_{j+1}^1 and Y_{j+1}^2 eventually:

$$H_j |Y_j^1\rangle_{A_{j+1}} |Y_j^2\rangle_{B_{j+1}} |\vec{\Gamma}_j\rangle_M = |Y_{j+1}^1\rangle_{A_{j+1}} |Y_{j+1}^2\rangle_{B_{j+1}} |\vec{\Gamma}_j\rangle_M. \quad (84)$$

This step can be accomplished with 5 addition and 2 addition-with-a-constant operations.

6. $T_j^\dagger$ uncomputes $|\vec{\Gamma}_j\rangle$ and restores register M back to the zero state. The step has the same cost as that of T_j .
7. $S_j^\dagger$ uncomputes $|\vec{\Lambda}_j\rangle$ and restores register L back to the zero state. The step has the same cost as that of S_j .
8. $R_j^\dagger$ uncomputes $|\sqrt{Y_j^2}\rangle$ and restores register J back to the zero state. The step has the same cost as that of R_j .

Overall, we have implemented V_j by using 5 addition, 2 addition-with-a-constant, 4 multiplication, 10 multiplication-with-a-constant, 2 square root operations, and Clifford gates, with the help of ancilla registers J, M, L (and other ancilla qubits necessary for performing the arithmetic operations).

U_1 for weak Euler scheme. The implementation of U_1 for weak Euler scheme is similar to the one for strong Euler scheme, except that a few steps are different (in fact, simpler) now.

We still decompose U_1 into $N + 2$ unitary operations:

$$U_1 = V_{N-1}V_{N-2} \dots V_1V_0Q_2Q_1. \quad (85)$$

Q_1 is the same as before, but now Q_2 loads the uniform distribution on $\{0, 1\}^{2N}$. So we simply have $Q_2 = H^{\otimes 2N}$ (each register E_j contains a single qubit now).

For $j = 0, 1, \dots, N - 1$, the unitary operator V_j still performs the transformation:

$$\begin{aligned} & V_j |Y_j^1\rangle_{A_j} |Y_j^2\rangle_{B_j} |\alpha_j\rangle_{E_{2j}} |\beta_j\rangle_{E_{2j+1}} |0\rangle_{A_{j+1}} |0\rangle_{B_{j+1}} \\ &= |Y_j^1\rangle_{A_j} |Y_j^2\rangle_{B_j} |\alpha_j\rangle_{E_{2j}} |\beta_j\rangle_{E_{2j+1}} |Y_{j+1}^1\rangle_{A_{j+1}} |Y_{j+1}^2\rangle_{B_{j+1}} \end{aligned} \quad (86)$$

Now we decompose V_j into 6 unitary operations:

$$V_j = R_j^\dagger T_j^\dagger H_j T_j R_j G_j. \quad (87)$$

These operations work as follows:

- G_j copies Y_j^1 and Y_j^2 to registers A_{j+1} and B_{j+1} , respectively:

$$G_j |Y_j^1\rangle_{A_j} |Y_j^2\rangle_{B_j} |0\rangle_{A_{j+1}} |0\rangle_{B_{j+1}} = |Y_j^1\rangle_{A_j} |Y_j^2\rangle_{B_j} |Y_j^1\rangle_{A_{j+1}} |Y_j^2\rangle_{B_{j+1}} \quad (88)$$

This step requires only CNOT gates.

- R_j computes the square root of Y_n^2 and saves it in an ancilla register J :

$$R_j |Y_n^2\rangle_{B_j} |0\rangle_J = |Y_n^2\rangle_{B_j} |\sqrt{Y_j^2}\rangle_J. \quad (89)$$

This step requires a square root operation.

- Let $\vec{\Gamma}_j = (-Y_j^2 \cdot \frac{h}{2}, \sqrt{Y_j^2} \cdot \rho\sqrt{h}, \sqrt{Y_j^2} \cdot \sqrt{h(1-\rho^2)}, -Y_j^2 \cdot \kappa h, \sqrt{Y_j^2} \cdot \xi\sqrt{h})$. Then T_j computes these terms and saves them in an ancilla register M :

$$T_j |Y_j^2\rangle_{B_j} |\sqrt{Y_j^2}\rangle_J |\vec{0}\rangle_M = |Y_j^2\rangle_{B_j} |\sqrt{Y_j^2}\rangle_J |\vec{\Gamma}_j\rangle_M \quad (90)$$

This step requires 5 multiplication-with-a-constant operations.

- Depending on the values of α_j and β_j (which are either 0 or 1), H_j adds/subtracts the terms in Γ_j to/from the values in registers A_{j+1} and B_{j+1} . It also adds the two constant terms rh , $\kappa\theta h$ to those quantities. Eventually, we obtain Y_{j+1}^1 and Y_{j+1}^2 in registers A_{j+1} and B_{j+1} , respectively:

$$\begin{aligned} & H_j |Y_j^1\rangle_{A_{j+1}} |Y_j^2\rangle_{B_{j+1}} |\alpha_j\rangle_{E_{2j}} |\beta_j\rangle_{E_{2j+1}} |\vec{\Gamma}_j\rangle_M \\ &= |Y_{j+1}^1\rangle_{A_{j+1}} |Y_{j+1}^2\rangle_{B_{j+1}} |\alpha_j\rangle_{E_{2j}} |\beta_j\rangle_{E_{2j+1}} |\vec{\Gamma}_j\rangle_M. \end{aligned} \quad (91)$$

This step can be accomplished with 5 addition and 2 addition-with-a-constant operations and Clifford gates.

- $T_j^\dagger$ uncomputes $|\vec{\Gamma}_j\rangle$ and restores register M back to the zero state. The step has the same cost as that of T_j .

- $R_j^\dagger$ uncomputes $|\sqrt{Y_j^2}\rangle$ and restores register J back to the zero state. The step has the same cost as that of R_j .

Overall, we have implemented V_j by using 5 addition, 2 addition-with-a-constant, 10 multiplication-with-a-constant, 2 square root operations, and Clifford gates, with the assistance of ancilla registers J and M (and other ancilla qubits necessary for performing the arithmetic operations).

Remark: In principle, the quantum circuit for each V_j does not need $R_j^\dagger$, $T_j^\dagger$ or $S_j^\dagger$ (if present) in the end. We put them there to release the ancilla registers J , M or L (if present) so that they can be reused in future iterations. If we eliminate some or all of these operations, then we would need a new batch of qubits in each iteration. Consequently, the time efficiency of the algorithm will improve while its space efficiency will worsen.

4.3 Implementing the operator U_2

The implementation of U_2 depends on the specific option under question. So we will discuss the cases of Asian and barrier options separately.

Asian option. Here we consider an Asian call. Asian puts can be handled similarly.

Suppose the Asian call option has strike price K and expiration time T . Then its payoff function is

$$f_0(\vec{S}) = \left(\frac{1}{N} \sum_{j=1}^N S_{jh} - K \right)^+. \quad (92)$$

Let Z be (an upper bound on) the maximum payoff of the option⁸. Then the normalized payoff function is

$$f(\vec{S}) = \frac{1}{Z} \left(\frac{1}{N} \sum_{j=1}^N S_{jh} - K \right)^+. \quad (93)$$

In terms of the log returns $\vec{R}$, the normalized payoff function becomes

$$\tilde{f}(\vec{R}) = \frac{1}{Z} \left(\frac{S_0}{N} \sum_{j=1}^N e^{R_{jh}} - K \right)^+ = \left(c \sum_{j=1}^N e^{R_{jh}} - k \right)^+, \quad (94)$$

where $c = S_0/(NZ)$ and $k = K/Z$ do not depend on the path.

For convenience, we define $X(\omega) = c \sum_{j=1}^N e^{Y_j^1(\omega)} - k$. Then $\tilde{f}(\vec{Y}^1(\omega)) = X^+(\omega)$. Our goal is to construct a unitary operation U_2 such that

$$U_2 \left| \vec{Y}^1(\omega) \right\rangle_A \left| 0 \right\rangle_C \left| 00 \dots 0 \right\rangle_G = \left| \vec{Y}^1(\omega) \right\rangle_A \left| X^+(\omega) \right\rangle_C \left| \Psi(\omega) \right\rangle_G, \quad \forall \omega \in \Omega, \quad (95)$$

⁸In practice, we estimate Z by simulating the stochastic process on a classical computer and generating multiple random paths and returning the maximum payoff among these paths. Suppose Z_M is the maximum payoff among M random paths. Then with high confidence, we conclude that the probability of a random path having payoff larger than Z_M is at most $O(1/M)$. By picking sufficiently large M , we can ensure that this probability is extremely low. Therefore, we could safely ignore the paths whose payoffs are larger than Z_M in our calculation.

where $|\Psi(\omega)\rangle$ is a normalized state depending on ω . This can be accomplished by concatenating three unitary operations:

$$U_2 = U_{2,3}U_{2,2}U_{2,1}, \quad (96)$$

where

$$U_{2,1} |Y_1^1, Y_2^1, \dots, Y_N^1\rangle_A |0, 0, \dots, 0\rangle_{G_1} = |Y_1^1, Y_2^1, \dots, Y_N^1\rangle_A |e^{Y_1^1}, e^{Y_2^1}, \dots, e^{Y_N^1}\rangle_{G_1}, \quad (97)$$

$$U_{2,2} |e^{Y_1^1}, e^{Y_2^1}, \dots, e^{Y_N^1}\rangle_{G_1} |0\rangle_{G_2} = |e^{Y_1^1}, e^{Y_2^1}, \dots, e^{Y_N^1}\rangle_{G_1} \left| c \sum_{j=1}^N e^{Y_j^1} - k \right\rangle_{G_2}, \quad (98)$$

$$U_{2,3} |X\rangle_{G_2} |0\rangle_C = |X\rangle_{G_2} |X^+\rangle_C. \quad (99)$$

That is, $U_{2,1}$ computes the exponentials of the Y_j^1 's, $U_{2,2}$ computes X from these exponentials, and $U_{2,3}$ computes X^+ from X . The registers G_1 and G_2 contain Nn and n qubits, respectively, and they form the subsystem G in Eq. (95).

Next, we analyze the costs of the three steps. $U_{2,1}$ requires N exponential operations. Meanwhile, $U_{2,2}$ can be implemented with $N - 1$ addition, one multiplication-with-a-constant and one subtraction-with-a-constant operations, and Clifford gates. Finally, $U_{2,3}$ can be implemented with n Toffoli gates and Clifford gates.

Barrier option. Here we consider an up-and-out put. The other types of barrier options can be handled similarly.

Suppose the up-and-out put option has strike price K , barrier B and expiration time T . Then its payoff function is

$$f_0(\vec{S}) = 1_{\max_{1 \leq j \leq N} S_{jh} \leq B} \cdot (K - S_T)^+. \quad (100)$$

Let Z be (an upper bound on) the maximum payoff of the option. Then the normalized payoff function is

$$f(\vec{S}) = 1_{\max_{1 \leq j \leq N} S_{jh} \leq B} \cdot \frac{(K - S_T)^+}{Z}. \quad (101)$$

In terms of the log returns $\vec{R}$, the normalized payoff function becomes

$$\tilde{f}(\vec{R}) = 1_{\max_{1 \leq j \leq N} R_{jh} \leq \ln(B/S_0)} \cdot \frac{(K - S_0 e^{R_T})^+}{Z} = 1_{\max_{1 \leq j \leq N} R_{jh} \leq \ln(B/S_0)} \cdot (k - c e^{R_T})^+ \quad (102)$$

where $c = S_0/Z$ and $k = K/Z$ do not depend on the path.

For convenience, we define $\eta_j(\omega) = 1$ if $Y_j^1(\omega) \leq \ln(B/S_0)$ and 0 otherwise, for $j = 1, 2, \dots, N$, and define $\eta(\omega) = \eta_1(\omega)\eta_2(\omega) \dots \eta_N(\omega)$. In addition, let $X(\omega) = k - c e^{Y_N^1(\omega)}$. Then we have $\tilde{f}(\vec{Y}^1(\omega)) = \eta(\omega)X^+(\omega)$.

Our goal is to construct a unitary operation U_2 such that

$$U_2 |\vec{Y}^1(\omega)\rangle_A |0\rangle_C |00 \dots 0\rangle_G = |\vec{Y}^1(\omega)\rangle_A |\eta(\omega)X^+(\omega)\rangle_C |\Psi(\omega)\rangle_G, \quad \forall \omega \in \Omega, \quad (103)$$

where $|\Psi(\omega)\rangle$ is a normalized state depending on ω . This can be accomplished by concatenating five unitary operations:

$$U_2 = U_{2,5}U_{2,4}U_{2,3}U_{2,2}U_{2,1} \quad (104)$$

where

$$U_{2,1} |Y_1^1, Y_2^1, \dots, Y_N^1\rangle_A |0, 0, \dots, 0\rangle_{G_1} = |Y_1^1, Y_2^1, \dots, Y_N^1\rangle_A |\eta_1, \eta_2, \dots, \eta_N\rangle_{G_1}, \quad (105)$$

$$U_{2,2} |\eta_1, \eta_2, \dots, \eta_N\rangle_{G_1} |0\rangle_{G_2} = |\eta_1, \eta_2, \dots, \eta_N\rangle_{G_1} |\eta_1 \eta_2 \dots \eta_N\rangle_{G_2}, \quad (106)$$

$$U_{2,3} |Y_N^1\rangle_{A_N} |0\rangle_{G_3} = |Y_N^1\rangle_{A_N} |k - ce^{Y_N^1}\rangle_{G_3}, \quad (107)$$

$$U_{2,4} |X\rangle_{G_3} |0\rangle_{G_4} = |X\rangle_{G_3} |X^+\rangle_{G_4}, \quad (108)$$

$$U_{2,5} |\eta\rangle_{G_2} |X^+\rangle_{G_4} |0\rangle_C = |\eta\rangle_{G_2} |X^+\rangle_{G_4} |\eta X^+\rangle_C. \quad (109)$$

That is, $U_{2,1}$ computes the η_j 's from the Y_j^1 's, $U_{2,2}$ computes η from the η_j 's, $U_{2,3}$ computes X from Y_N^1 , $U_{2,4}$ computes X^+ from X , and $U_{2,5}$ computes ηX^+ from η and X^+ . The registers G_1 , G_2 , G_3 and G_4 contain N , 1, n and n qubits, respectively, and they form the subsystem G in Eq. (103).

The costs of the five steps are as follows. $U_{2,1}$ requires N comparison (with a constant) operations. $U_{2,2}$ can be implemented as an $(N + 1)$ -qubit Toffoli gate. $U_{2,3}$ requires one exponential, one multiplication-with-a-constant, and one addition-with-a-constant operations. $U_{2,4}$ can be implemented with n Toffoli gates and Clifford gates. Finally, $U_{2,5}$ can be implemented with n Toffoli gates.

4.4 Implementing the operator U_3

The operator U_3 appears frequently in the literature (e.g. [2, 6, 36]) and we adopt a common method to realize it. Specifically, we decompose U_3 into two unitary operations:

$$U_3 = U_{3,2} U_{3,1}, \quad (110)$$

where

$$U_{3,1} |z\rangle_C |0\rangle_H = |z\rangle_C |\arcsin(\sqrt{z})\rangle_H, \quad (111)$$

$$U_{3,2} |\arcsin(\sqrt{z})\rangle_H |0\rangle_D = |\arcsin(\sqrt{z})\rangle_H [\sqrt{z}|0\rangle_D + \sqrt{1-z}|1\rangle_D], \quad (112)$$

for all $z \in [0, 1]$. Here the register H contains n qubits. We implement $U_{3,1}$ as an arcsin-of-square-root operation defined in Appendix A. Meanwhile, we implement $U_{3,2}$ as a $U_{\sin}$ operation defined in Appendix B.

4.5 Resource analysis

So far, we have constructed the circuit $\mathcal{A}$ which satisfies Eq. (62) by composing the modules in Appendices A, B and C as well as Clifford gates. Suppose it contains $m + 1$ qubits. Our algorithm makes N_{oracle} uses of the unitary operation $Q = \mathcal{A}R_0\mathcal{A}^{-1}S_0$, where $R_0 = I_{m+1} - 2|0^{m+1}\rangle\langle 0^{m+1}|$ can be implemented with an $(m + 1)$ -qubit Toffoli gate and Clifford gates, and $S_0 = -Z \otimes I_m$ costs no T gate. So the total number of T gates in our algorithm is

$$N_{\text{oracle}} \text{T}_{\text{count}}(Q) = N_{\text{oracle}} [2\text{T}_{\text{count}}(\mathcal{A}) + \text{T}_{\text{count}}(\text{Toffoli}_{m+1})]. \quad (113)$$

In Appendices A, B and C, we introduce the fixed-point quantum arithmetic operations, a block-encoding $U_{\sin}$ of the sine function, and a unitary operation U_{gauss} for preparing Gaussian states, respectively, and analyze the costs of implementing these operations. From now on, we will adopt the notation in these appendices.

The number of T gates in $\mathcal{A}$ is

$$T_{\text{count}}(\mathcal{A}) = T_{\text{count}}(U_1) + T_{\text{count}}(U_2) + T_{\text{count}}(U_3) \quad (114)$$

where U_1 , U_2 and U_3 satisfy Eqs. (66), (67) and (68) respectively.

The number of T gates in U_1 depends on which numerical scheme we use. For strong Euler method, it is

$$T_{\text{count}}(U_1) = T_{\text{count}}(Q_1) + T_{\text{count}}(Q_2) + \sum_{j=0}^{N-1} T_{\text{count}}(V_j) \quad (115)$$

$$\begin{aligned} &= N[2T_{\text{count}}(U_{\text{gauss}}(n, \eta, \epsilon_{\text{prep}}, \epsilon_{\text{gauss}})) + 5T_{\text{count}}(\text{ADD}_n) \\ &\quad + 2T_{\text{count}}(\text{ADD_CONST}_n) + 4T_{\text{count}}(\text{MUL}_{n,p}) \\ &\quad + 10T_{\text{count}}(\text{MUL_CONST}_{n,p}) + 2T_{\text{count}}(\text{SQRT}_n)]. \end{aligned} \quad (116)$$

By contrast, for weak Euler method, it is

$$T_{\text{count}}(U_1) = T_{\text{count}}(Q_1) + T_{\text{count}}(Q_2) + \sum_{j=0}^{N-1} T_{\text{count}}(V_j) \quad (117)$$

$$\begin{aligned} &= N[5T_{\text{count}}(\text{ADD}_n) + 2T_{\text{count}}(\text{ADD_CONST}_n) \\ &\quad + 10T_{\text{count}}(\text{MUL_CONST}_{n,p}) + 2T_{\text{count}}(\text{SQRT}_n)], \end{aligned} \quad (118)$$

which is much smaller, as will be shown in Section 4.7.

The number of T gates in U_2 depends on the type of the option under consideration. For example, for an Asian call, it is

$$T_{\text{count}}(U_2) = T_{\text{count}}(U_{2,1}) + T_{\text{count}}(U_{2,2}) + T_{\text{count}}(U_{2,3}) \quad (119)$$

$$\begin{aligned} &= NT_{\text{count}}(\text{EXP}_{n,p,\epsilon_{\text{exp}}}) + (N-1)T_{\text{count}}(\text{ADD}_n) \\ &\quad + T_{\text{count}}(\text{MUL_CONST}_{n,p}) + T_{\text{count}}(\text{SUB_CONST}_n) + nT_{\text{count}}(\text{Toffoli}_3). \end{aligned} \quad (120)$$

Meanwhile, for an up-and-out put, it is

$$T_{\text{count}}(U_2) = T_{\text{count}}(U_{2,1}) + T_{\text{count}}(U_{2,2}) + T_{\text{count}}(U_{2,3}) + T_{\text{count}}(U_{2,4}) + T_{\text{count}}(U_{2,5}) \quad (121)$$

$$\begin{aligned} &= NT_{\text{count}}(\text{COMP_CONST}_n) + T_{\text{count}}(\text{Toffoli}_{N+1}) + T_{\text{count}}(\text{EXP}_{n,p,\epsilon_{\text{exp}}}) \\ &\quad + T_{\text{count}}(\text{MUL_CONST}_{n,p}) + T_{\text{count}}(\text{ADD_CONST}_n) + 2nT_{\text{count}}(\text{Toffoli}_3). \end{aligned} \quad (122)$$

Finally, the number of T gates in U_3 is

$$T_{\text{count}}(U_3) = T_{\text{count}}(U_{3,1}) + T_{\text{count}}(U_{3,2}) \quad (123)$$

$$= T_{\text{count}}(\text{ARCSIN_SQRT}_{n,p,\epsilon_{\text{arcsin}}}) + T_{\text{count}}(U_{\text{sin}}(n, \epsilon_{\text{sin}})). \quad (124)$$

To obtain the T-depth of our algorithm, we replace T_{count} with T_{depth} in the above equations. Note that this yields a conservative estimate of the necessary T-depth, because we could in principle improve it by running some modules, e.g., Q_1 and Q_2 , in parallel. But this kind of optimization is unlikely to significantly reduce the T-depth of the circuit, given the highly-sequential nature of our algorithm. Thus, we will not pursue this direction here.

To determine the number of qubits in the circuit $\mathcal{A}$, we scan through the operations in it, check the number of qubits necessary to store the permanent and temporary information and perform the current operation at each moment, and take the maximum among them. The number of qubits in Q is larger than that of $\mathcal{A}$ by 1.

4.6 Error analysis

There are multiple sources of errors in our algorithm:

1. Amplitude estimation can only produce an estimate of the true value with certain accuracy and confidence.
2. The results of fixed-point arithmetic operations contain errors due to the finite-precision representation of real numbers and/or the piecewise-polynomial approximation of exponential and arcsin functions in their implementation (see Appendix A for more details).
3. We use the circuit in Appendix B to implement U_{sin} . This circuit contains $R_x/R_y/R_z$ rotations which can only be synthesized approximately with Clifford and T gates. In this work, we utilize the method of [37] to synthesize $R_x/R_y/R_z$ rotations.
4. For a similar reason, U_{gauss} cannot be implemented exactly either.
5. In our algorithm, each α_j and β_j can only take values from a finite set of numbers. This poses a problem for strong Euler method, as it changes the probability distribution of the random path. Consequently, the expectation of the payoff in the simulation is different from the ideal one.
6. Finally, even if U_{gauss} is implemented perfectly, it only approximately prepares a quantum state that encodes a discrete version of standard normal distribution (see Appendix C for more details).

Note that the last three issues are specific to strong Euler method. They do not exist for weak Euler method.

Next, we analyze each of the above errors individually:

- The first error can be bounded analytically, and Eq. (64) gives an upper bound on the number of oracle queries needed to achieve the desired accuracy and confidence in amplitude estimation.
- The third error can be bounded as follows. Suppose each U_{sin} is implemented with ϵ_{sin} precision. Then since there are $2N_{\text{oracle}}$ such operations in our circuit, the output state of the circuit is at most $2N_{\text{oracle}}\epsilon_{\text{sin}}$ -away (in trace distance) from the ideal state, which means that the deviation of the final result due to this factor is at most $2N_{\text{oracle}}\epsilon_{\text{sin}}$ too.
- Similarly, if strong Euler method is employed and each U_{gauss} is implemented with precision ϵ_{gauss} , then the fourth error is at most $4NN_{\text{oracle}}\epsilon_{\text{gauss}}$.
- The fifth error can be also bounded easily. In the algorithm based on strong Euler method, we need $4NN_{\text{oracle}}$ copies of the state encoding a discrete version of standard normal distribution. Suppose each of them is prepared with ϵ_{prep} accuracy in trace distance. Then this leads to an error of at most $4NN_{\text{oracle}}\epsilon_{\text{prep}}$ in the final result.
- In principle, we could derive a theoretical upper bound on the error incurred by imperfect fixed-point arithmetic operations, as done in [6]. However, we find that such a bound is quite loose, as it always assumes the worst scenario, i.e., each arithmetic operation contains the largest possible error, and the error of a sequence of arithmetic operations is the sum of these individual errors, which almost never happens.

To better bound this error in practical situations, we simulate the stochastic process using strong/weak Euler method *under fixed-point arithmetic conditions* on a classical computer, generate multiple random paths, and compute the average distance between the true and calculated payoffs of the option for these paths. This procedure turns out to be highly efficient ⁹.

- Moreover, if the algorithm is based on strong Euler method, then we choose each α_j and β_j from a discrete version of standard normal distribution in the above simulation. Consequently, the estimated error will be the combined effect of flawed fixed-point arithmetic operations and discrete approximation of Gaussian distributions.

To summarize, if the algorithm is based on strong Euler method, then the error in our final estimate of $\mathbb{E} [\tilde{f}(\vec{Y}^1(\omega))]$ is at most

$$\epsilon_{\text{estimate}} + \epsilon_{\text{arithm/disc}} + 2N_{\text{oracle}}\epsilon_{\text{sin}} + 4NN_{\text{oracle}}(\epsilon_{\text{prep}} + \epsilon_{\text{gauss}}) \quad (125)$$

where

- $\epsilon_{\text{estimate}}$ is the precision of amplitude estimation,
- $\epsilon_{\text{arithm/disc}}$ is the error due to imperfect fixed-point arithmetic operations and discrete approximation of Gaussian distributions,
- ϵ_{sin} is the precision of implementing each U_{sin} ,
- ϵ_{prep} be the precision of preparing each Gaussian state,
- ϵ_{gauss} be the precision of implementing each U_{gauss} ,

whereas if the algorithm is based on weak Euler method, then the final error is at most

$$\epsilon_{\text{estimate}} + \epsilon_{\text{arithm}} + 2N_{\text{oracle}}\epsilon_{\text{sin}} \quad (126)$$

where

- $\epsilon_{\text{estimate}}$ is the precision of amplitude estimation,
- ϵ_{arithm} is the error due to imperfect fixed-point arithmetic operations,
- ϵ_{sin} is the precision of implementing each U_{sin} .

4.7 Case studies

Next, we apply our algorithms to four instances of option pricing under the Heston model to test their performance in practice. Specifically, we consider four settings of the Heston model which are described in Table 3, and estimate the value of an Asian or barrier option under each setting. The specifications of the options are listed in Table 4.

For each problem instance, we develop the quantum algorithms based on strong and weak Euler schemes, and tune their parameters, e.g., ϵ_{sin} , ϵ_{gauss} , ϵ_{prep} , to make sure that these two algorithms achieve (roughly) the same accuracy and confidence in their results. The specific settings of the parameters are given in Table 5. Note that the weak-Euler-based algorithm uses fewer qubits to represent each real number and implements each U_{sin}

Setting	r	ρ	κ	θ	ξ	S_0	ν_0
No. 1	0.03	-0.1	2	0.12	0.3	100	0.1
No. 2	0.05	-0.1	2	0.04	0.2	100	0.05
No. 3	0.03	0	2	0.03	0.2	100	0.03
No. 4	0.05	-0.1	2	0.09	0.2	100	0.06

Table 3: The settings of the Heston model studied in Section 4.7. Each setting includes the model parameters r , ρ , κ , θ , ξ and initial condition S_0 and ν_0 .

Instance	Setting	Option type	K	B	T	Z
No. 1	No. 1	Asian call	90	N/A	1.0	200
No. 2	No. 2	Asian put	110	N/A	1.0	100
No. 3	No. 3	Up-and-out call	90	170	1.0	80
No. 4	No. 4	Down-and-in put	110	80	1.0	100

Table 4: The problem instances studied in Section 4.7. The specification of each instance includes the setting of the Heston model and the type, strike K , barrier B (if needed) and expiration time T of the option. Here we also present an upper bound Z on the maximum payoff of the option which is used to normalize the payoff function.

Instance	Numerical Scheme	N	n	p	$\epsilon_{\sin}$	ϵ_{gauss}	ϵ_{prep}	$\epsilon_{\text{estimate}}$
No. 1	Strong Euler	256	29	11	10^{-9}	10^{-12}	10^{-12}	10^{-3}
No. 1	Weak Euler	256	27	11	10^{-8}	N/A	N/A	10^{-3}
No. 2	Strong Euler	256	29	10	10^{-9}	5×10^{-12}	5×10^{-12}	10^{-3}
No. 2	Weak Euler	256	27	10	10^{-8}	N/A	N/A	10^{-3}
No. 3	Strong Euler	1024	32	10	10^{-9}	5×10^{-13}	5×10^{-13}	10^{-3}
No. 3	Weak Euler	1024	29	10	5×10^{-9}	N/A	N/A	10^{-3}
No. 4	Strong Euler	1024	30	10	10^{-9}	10^{-12}	10^{-12}	10^{-3}
No. 4	Weak Euler	1024	29	10	5×10^{-9}	N/A	N/A	10^{-3}

Table 5: The parameter settings in our experiments, including the underlying numerical scheme, the number N of time steps, the parameters n and p about the fixed-point representation of real numbers, the precision $\epsilon_{\sin}$ of implementing each $U_{\sin}$, the precision ϵ_{gauss} of implementing each U_{gauss} , the precision ϵ_{prep} of preparing each Gaussian state, and the accuracy $\epsilon_{\text{estimate}}$ of amplitude estimation. Moreover, we set $\eta = 6.0$ in all Gaussian state preparation procedures, and set $\epsilon_{\text{exp}} = \epsilon_{\arcsin} = 10^{-6}$, and set the failure probability of amplitude estimation to $\delta = 0.1$ in all experiments.

with lower precision than the strong-Euler-based algorithm, yet it still produces as accurate result as the latter.

Table 6 summarizes the resources required by the two quantum algorithms to reach the same accuracy and confidence on each problem instance.

Instance	Numerical Scheme	N	Error	T-count	T-depth	Number of Logical Qubits
No. 1	Strong Euler	256	1.3×10^{-3}	4.1×10^{13}	2.9×10^{13}	3.8×10^4
No. 1	Weak Euler	256	1.3×10^{-3}	2.4×10^{11}	1.2×10^{11}	2.2×10^4
No. 2	Strong Euler	256	1.2×10^{-3}	3.9×10^{13}	2.7×10^{13}	3.8×10^4
No. 2	Weak Euler	256	1.2×10^{-3}	2.3×10^{11}	1.1×10^{11}	2.2×10^4
No. 3	Strong Euler	1024	1.3×10^{-3}	1.7×10^{14}	1.2×10^{14}	1.3×10^5
No. 3	Weak Euler	1024	1.3×10^{-3}	4.4×10^{11}	2.2×10^{11}	6.4×10^4
No. 4	Strong Euler	1024	1.4×10^{-3}	1.7×10^{14}	1.2×10^{14}	1.3×10^5
No. 4	Weak Euler	1024	1.4×10^{-3}	4.4×10^{11}	2.2×10^{11}	6.4×10^4

Table 6: The resources required by strong- and weak-Euler-based algorithms to achieve the same accuracy and confidence on each problem instance, including the T-counts, T-depths and numbers of logical qubits in the circuits. Note that the weak-Euler-based algorithm is far more efficient than the strong-Euler-based algorithm on every instance. Here the target quantity is $\mathbb{E} [\tilde{f}(\tilde{Y}^1(\omega))]$ for the corresponding numerical scheme with N time steps.

One can see that the algorithm based on weak Euler scheme has much smaller T-count and T-depth than the one based on strong Euler scheme, and uses fewer qubits as well. In fact, the strong-Euler-based algorithm demands hundreds of times more T gates than the weak-Euler-based algorithm on all instances. This is mainly due to the expensive procedure of preparing the Gaussian states needed by the former. Specifically, Table 7 shows the T-counts of the operations U_1 , U_2 , U_3 and Q as well as the number of applications of Q in each algorithm on each instance. Note that the strong-Euler-based algorithm has much more costly U_1 than the weak-Euler-based algorithm. We find that in the this algorithm, $Q_2 = U_{\text{gauss}}^{\otimes 2N}$ dominates the consumption of T gates within U_1 , i.e., $T_{\text{count}}(Q_2) \geq 0.99T_{\text{count}}(U_1)$. This is mostly because U_{gauss} contains a large number, i.e., $> 10^5$, of R_y/R_z rotations, each of which needs to be synthesized with Clifford and T gates with high precision. On the other hand, the weak-Euler-based algorithm does not need to prepare Gaussian states and hence simulates the stochastic process with much lower costs.

We emphasize that the above resource estimates hold under the experimental settings delineated in Tables 3, 4 and 5 only. Nevertheless, the conclusion that the algorithm based on weak Euler scheme is far superior to the one based on strong Euler scheme remains valid in general.

5 Conclusion and outlook

To summarize, we have developed quantum algorithms for pricing Asian and barrier options under the Heston model, and estimated their costs and errors on example instances under typical market conditions. These algorithms are based on combining classical numerical methods for stochastic differential equations and quantum amplitude estimation

⁹Here we aim to estimate the difference between the payoffs of the option for two random paths that are always close to each other. The variance of this random variable is quite small. Thus, it takes few samples to reach high accuracy in this estimation.

Instance	Numerical Scheme	$T_{\text{count}}(U_1)$	$T_{\text{count}}(U_2)$	$T_{\text{count}}(U_3)$	$T_{\text{count}}(Q)$	N_{oracle}
No. 1	Strong Euler	2.8×10^9	1.0×10^7	7.0×10^4	5.6×10^9	7363
No. 1	Weak Euler	6.4×10^6	9.3×10^6	6.0×10^4	3.2×10^7	7363
No. 2	Strong Euler	2.6×10^9	1.0×10^7	6.8×10^4	5.3×10^9	7363
No. 2	Weak Euler	6.4×10^6	9.2×10^6	6.0×10^4	3.2×10^7	7363
No. 3	Strong Euler	1.2×10^{10}	3.0×10^5	8.0×10^4	2.4×10^{10}	7363
No. 3	Weak Euler	2.9×10^7	2.7×10^5	6.8×10^4	6.0×10^7	7363
No. 4	Strong Euler	1.1×10^{10}	2.8×10^5	7.2×10^4	2.3×10^{10}	7363
No. 4	Weak Euler	2.9×10^7	2.7×10^5	6.8×10^4	6.0×10^7	7363

Table 7: The costs of implementing U_1 , U_2 , U_3 , Q and the number of applications of Q in each algorithm on each instance.

technique. In particular, we empirically showed that, despite its simplicity, weak Euler method achieves the same level of accuracy as strong Euler method for option pricing under the Heston model. Furthermore, by eliminating the expensive procedure of preparing Gaussian states, the quantum algorithm based on weak Euler scheme achieves drastically better efficiency than the one based on strong Euler scheme. Our results shed light on the possibility of using quantum computers to accelerate option pricing under stochastic volatility in the future.

Even though our quantum algorithms achieve only a quadratic speedup over classical Monte Carlo methods, they avoid several pitfalls that can make many quantum algorithms fast in theory but potentially slow in practice. First, our algorithms do not require quantum random access memory (QRAM), which could be difficult to implement. Instead, we receive the problem specification in classical form and then build and execute the quantum circuits. Second, our algorithms directly produce the estimate of the target quantity, unlike other approaches—such as those based on quantum linear system solvers [36]—which must first generate a quantum state encoding the solution and then repeatedly measure it to obtain the final result, leading to additional overhead. Third, our numerical results suggest that the costs of our algorithms are relatively insensitive to the problem specification. Their performance does not rely on assumptions like the well-conditioning of a data matrix, which could be difficult to satisfy in practice. Additionally, our algorithms use computational resources similar to or fewer than those required for many quantum chemistry simulations [38, 39, 40, 41]. For these reasons, we believe that option pricing under stochastic volatility could be among the first useful applications of quantum computers.

To our knowledge, this is the first work on fault-tolerant quantum algorithms with gate-by-gate level instructions and resource estimates for pricing exotic options under a stochastic volatility model. Previous works on derivative pricing have been restricted to the case that the asset price follows a geometric Brownian motion with constant drift and volatility. Due to different problem settings, it is hard to fairly compare our results and previous results. Among the previous works, [6] is most similar to ours and considers the pricing of autocallable and Target Accrual Redemption Forward (TARF) derivatives in the GBM case. In their problems, they have 20 time steps, while we have over 200 or 1000 time steps here. If one compares the average number of T gates per time step, then our cost is similar to theirs, even though we consider a more complicated model which requires more arithmetic operations to simulate. Moreover, [6] uses strong Euler method to simulate their stochastic process and thus needs to prepare Gaussian states (which is done heuristically using a variational algorithm in their work). We believe their results

can quite possibly be improved by utilizing weak Euler method instead.

Our resource estimates can be used to derive the requirement for quantum hardware to demonstrate practical advantage for option pricing under stochastic volatility. Assuming a target of $\sim 10^3$ seconds for pricing a barrier option, the quantum processor would need to execute each layer of T gates at a rate of $\sim 10\text{MHz}$, i.e., logical clockspeed, where each layer consists of ~ 10 T gates. While [6] also concludes that $\sim 10\text{MHz}$ per layer of T gate is needed for their algorithms to provide practical advantage, their algorithms have $\sim 10^3$ T gates per layer, which, compared to ~ 10 T gates per layer, is more difficult to apply in parallel at a high rate, especially in early fault tolerant devices with limited number of magic state factories. In this sense, our hardware requirement is less stringent than theirs. We stress that in practice, the simulation duration for option pricing under stochastic volatility could be longer than 10^3 seconds if a more accurate pricing, i.e., smaller ϵ , is desired, in which case the hardware threshold is further relaxed because the quantum computational costs grow more slowly than the classical costs.

There are multiple research directions that deserve further exploration:

- So far, we have employed strong/weak Euler method for SDEs to simulate the dynamics of the Heston model. As a consequence, our algorithm has time cost proportional to the evolution time T . Is it possible to utilize more sophisticated strategy, e.g., spectral methods [8], to develop quantum algorithms for the same problem with only poly($\log(T)$) runtime?
- Our algorithm consists of a large number of arithmetic operations which become a main bottleneck of the algorithm. It would be beneficial to investigate whether alternative techniques similar to those in [9] are applicable to stochastic volatility models.
- Besides Monte Carlo simulation, another approach to pricing options is to reduce the problem to a PDE to be solved numerically [10]. It would be interesting to analyze the cost of the quantum version of this approach and compare it with ours.
- Can we develop efficient quantum algorithms for pricing American options under stochastic volatility? This might require quantization of the binomial tree method [11] or generalization of the results in [42].
- Recently, there has been progress in the development of low-depth amplitude estimation algorithms [33, 43, 34] which are more suitable for early fault-tolerant quantum computers [44, 45, 46, 47, 48, 49]. It would be interesting to apply these algorithms to option pricing under stochastic volatility models and see how the cost would change in the depth-limited setting. This would help us understand the potential of achieving quantum advantage for this problem in the near future.

A Fixed-point quantum arithmetic operations

In this appendix, we describe the elementary quantum arithmetic operations used in our algorithm, and analyze the costs of implementing them.

We use two's complement to represent real numbers in our circuits. That is, every number is represented by an n -bit string in which the most significant bit indicates the sign of the number, the next $p - 1$ bits are used to denote the integral part, and the other $n - p$ bits are used to denote the fractional part. Here n and p are fixed throughout the computation.

Formally, if a real number x has format $x_{n-1}x_{n-2}\dots x_{n-p}.x_{n-p-1}\dots x_0$, where $x_i \in \{0,1\}$ for each i , then it has the value

$$x = -x_{n-1}2^{p-1} + x_{n-2}2^{p-2} + \dots + x_{n-p}2^0 + x_{n-p-1}2^{-1} + \dots + x_02^{-n+p}. \quad (127)$$

Note that if $x_{n-1} = 1$, then x is negative; otherwise, x is positive or 0. This number is represented by the state $|x_{n-1}, x_{n-2}, \dots, x_0\rangle$ on a quantum computer.

Let $S_{n,p} = \{z2^{-n+p} : z \in [-2^{n-1}, 2^{n-1}] \cap \mathbb{Z}\}$ be the set of real numbers that can be written as in Eq. (127). Note that $S_{n,p} \subset [-2^{p-1}, 2^{p-1}]$. Moreover, for any $x \in [-2^{p-1}, 2^{p-1}]$, let $\bar{x} = 2^{p-n} \lfloor 2^{n-p}x \rfloor$ be the largest number in $S_{n,p}$ that does not exceed x . This number is at most 2^{p-n} -away from x .

We choose sufficiently large p so that overflow never happens in our algorithm. That is, e.g., if we need to compute $x + y$ somewhere, then $|x + y| < 2^{p-1}$, and similarly for other arithmetic operations. This is a necessary condition to ensure that the fixed-point representation remains valid for the entire computation.

Multiqubit Toffoli gates. The n -qubit Toffoli gate is defined as follows:

$$\text{Toffoli}_n : |x_1, x_2, \dots, x_{n-1}\rangle |x_n\rangle \rightarrow |x_1, x_2, \dots, x_{n-1}\rangle |x_n \oplus x_1x_2 \dots x_{n-1}\rangle, \quad (128)$$

for all $x_1, x_2, \dots, x_n \in \{0,1\}^n$. It plays a crucial role in realizing logical AND and OR operations as well as the reflection operator $R_0 = I - 2|00\dots 0\rangle\langle 00\dots 0|$ in amplitude estimation.

We utilize two different methods to implement this gate depending on the scenario:

- The method of [50] achieves $4(n-2)$ T-count and $n-2$ T-depth, but requires $n-1$ ancilla qubits. We employ it to implement most of the multiqubit Toffoli gates in our algorithm, as it is more time-efficient.
- For $n \geq 5$, the method of [51] needs a single ancilla qubit, but has $16n-60$ T-count and at most $16n-60$ T-depth. We use it to implement the multiqubit Toffoli gates related to R_0 , as it is more space-efficient (and we do not run many R_0 's in our algorithm).

Addition/Subtraction. The quantum adder and subtractor are defined as follows:

$$\text{ADD}_n : |x\rangle |y\rangle \rightarrow |x\rangle |x+y\rangle, \quad (129)$$

$$\text{SUB}_n : |x\rangle |y\rangle \rightarrow |x\rangle |x-y\rangle, \quad (130)$$

for $x, y \in S_{n,p}$ such that $|x \pm y| < 2^{p-1}$. We utilize the method of [52] to implement these operations which has $4n-4$ T-count and $2n-2$ T-depth and requires $n-1$ ancilla qubits.

Sometimes we only need to add/subtract a known constant to/from the number in a register. So it is useful to introduce the following variant of quantum adder and subtractor:

$$\text{ADD_CONST}_n(c) : |x\rangle \rightarrow |x+c\rangle, \quad (131)$$

$$\text{SUB_CONST}_n(c) : |x\rangle \rightarrow |x-c\rangle, \quad (132)$$

for $x \in S_{n,p}$ such that $|x \pm c| < 2^{p-1}$, for given $c \in S_{n,p}$. These operations can be implemented by a variant of the method of [52] which achieves $4n-8$ T-count and $2n-4$ T-depth and demands $2n-2$ ancilla qubits.

In addition, our algorithm also needs controlled-adders and controlled-subtractors which are defined as follows:

$$c - \text{ADD}_n : |a\rangle |x\rangle |y\rangle \rightarrow \begin{cases} |a\rangle |x\rangle |x+y\rangle, & \text{if } a = 1, \\ |a\rangle |x\rangle |y\rangle, & \text{otherwise,} \end{cases} \quad (133)$$

$$c - \text{SUB}_n : |a\rangle |x\rangle |y\rangle \rightarrow \begin{cases} |a\rangle |x\rangle |x-y\rangle, & \text{if } a = 1, \\ |a\rangle |x\rangle |y\rangle, & \text{otherwise,} \end{cases} \quad (134)$$

for $a \in \{0, 1\}$ and $x, y \in S_{n,p}$ such that $|x \pm y| < 2^{p-1}$ if $a = 1$. One can use the method of [52] to implement these operations in $8n - 4$ T-count and $4n - 2$ T-depth, with the help of $2n - 1$ ancilla qubits.

Comparison. We often need to determine whether the number in a register exceeds a threshold or not. To this end, we introduce the following quantum comparator:

$$\text{COMP_CONST}_n(c) : |x\rangle |0\rangle \rightarrow \begin{cases} |x\rangle |1\rangle, & \text{if } x \geq c, \\ |x\rangle |0\rangle, & \text{otherwise,} \end{cases} \quad (135)$$

for all $x \in S_{n,p}$, for given $c \in S_{n,p}$. This operation can be realized by computing $x - c$, copying its sign bit, and uncomputing $x - c$, i.e.,

$$\begin{aligned} |x\rangle |0\rangle |0\rangle &\rightarrow |x\rangle |x\rangle |0\rangle \\ &\rightarrow |x\rangle |x - c\rangle |0\rangle \\ &\rightarrow |x\rangle |x - c\rangle |\text{sgn}(x - c)\rangle \\ &\rightarrow |x\rangle |x\rangle |\text{sgn}(x - c)\rangle \\ &\rightarrow |x\rangle |0\rangle |\text{sgn}(x - c)\rangle, \end{aligned} \quad (136)$$

where the first, third and last steps require only CNOT gates, the second and fourth steps require $\text{SUB_CONST}_n(c)$ and its inverse. Thus, we can implement $\text{COMP_CONST}_n(c)$ in $8n - 16$ T-count and $4n - 8$ T-depth, with the assistance of $3n - 2$ ancilla qubits.

Multiplication. The quantum multiplier is defined as follows:

$$\text{MUL}_{n,p} : |x\rangle |y\rangle |0\rangle \rightarrow |x\rangle |y\rangle |\overline{xy}\rangle, \quad (137)$$

for $x, y \in S_{n,p}$ such that $|xy| < 2^{p-1}$. Using standard shift-and-add strategy, one can be implement this operation as a sequence of controlled-adders. The number of T gates in the resulting circuit is

$$T_{\text{count}}(\text{MUL}_{n,p}) = \sum_{i=0}^{p-1} T_{\text{count}}(c - \text{ADD}_{n-i}) + \sum_{i=1}^{n-p} T_{\text{count}}(c - \text{ADD}_{n-i}) \quad (138)$$

$$= \sum_{i=0}^{p-1} [8(n-i) - 4] + \sum_{i=1}^{n-p} [8(n-i) - 4] \quad (139)$$

$$= 4n^2 - 8n + 8pn - 8p^2 + 8p. \quad (140)$$

Furthermore, this circuit has T-depth $2n^2 - 4n + 4pn - 4p^2 + 4p$, and requires $2n - 1$ ancilla qubits.

Sometimes we only need to multiply the number in a register by a known constant. Therefore, it is useful to introduce the following variant of quantum multiplier:

$$\text{MUL_CONST}_{n,p}(c) : |x\rangle |0\rangle \rightarrow |x\rangle |\overline{cx}\rangle, \quad (141)$$

for $x \in S_{n,p}$ such that $|cx| < 2^{p-1}$, for given $c \in S_{n,p}$. This operation can be implemented as a sequence of adders. The number of T gates in the resulting circuit is

$$\text{T}_{\text{count}}(\text{MUL_CONST}_{n,p}) = \sum_{i=0}^{p-1} \text{T}_{\text{count}}(\text{ADD}_{n-i}) + \sum_{i=1}^{n-p} \text{T}_{\text{count}}(\text{ADD}_{n-i}) \quad (142)$$

$$= \sum_{i=0}^{p-1} [4(n-i) - 4] + \sum_{i=1}^{n-p} [4(n-i) - 4] \quad (143)$$

$$= 2n^2 - 6n + 4pn - 4p^2 + 4p. \quad (144)$$

Moreover, this circuit has T-depth $n^2 - 3n + 2pn - 2p^2 + 2p$, and uses $n - 1$ ancilla qubits.

Square Root. The quantum version of square root is defined as:

$$\text{SQRT}_n : |x\rangle |0\rangle \rightarrow |x\rangle |\sqrt{x}\rangle, \quad (145)$$

for all $x \in S_{n,p} \cap \mathbb{R}^{\geq 0}$. [53] gives a strategy for implementing a similar operation which maps $|x\rangle |0\rangle$ to $|Z_x\rangle |\sqrt{x}\rangle$, where Z_x depends on x . We change it to a clean version by the do-copy-undo trick, and also replace its adders by the ones of [52] which cost fewer T gates. The resulting circuit has

$$\text{T}_{\text{count}}(\text{SQRT}_n) = 2 \sum_{i=1}^{\lceil n/2 \rceil} \text{T}_{\text{count}}(\text{ADD}_{2i}) + 2\text{T}_{\text{count}}(c - \text{ADD}_{2\lceil n/2 \rceil}) \quad (146)$$

$$= 2 \sum_{i=1}^{\lceil n/2 \rceil} (8i - 4) + 2(16\lceil n/2 \rceil - 4) \quad (147)$$

$$= 8\lceil n/2 \rceil^2 + 32\lceil n/2 \rceil - 8 \quad (148)$$

T gates. Furthermore, it has T-depth $4\lceil n/2 \rceil^2 + 16\lceil n/2 \rceil - 4$ and uses $\lceil 3.5n \rceil$ ancilla qubits.

Piecewise Polynomial. A function g on a domain $[a, b]$ is called an (M, d) -piecewise polynomial if we can partition $[a, b]$ into M subintervals $[a_0, a_1), [a_1, a_2), \dots, [a_{M-1}, a_M]$ (where $a = a_0 < a_1 < \dots < a_M = b$) such that g equals a degree- d polynomial p_i on the i -th subinterval, i.e., $g(x) = p_i(x)$ for all $x \in [a_{i-1}, a_i)$, for $i = 1, 2, \dots, M$. Piecewise polynomials are interesting because many natural functions, e.g., $\tanh(x)$, $\sin(x)$, $\exp(-x^2)$, can be well-approximated by them with small numbers of subintervals or low polynomial degrees.

Let pp be an (M, d) -piecewise polynomial. We define its quantum version as:

$$\text{PPoly}_{n,p,M,d} : |x\rangle |0\rangle \rightarrow |x\rangle |\overline{pp(x)}\rangle \quad (149)$$

for $x \in S_{n,p}$ such that $|pp(x)| < 2^{p-1}$. [54] develops a method to implement a similar operation which maps $|x\rangle |0\rangle |0\rangle$ to $|x\rangle |\overline{pp(x)}\rangle |\Gamma_x\rangle$, where Γ_x depends on x . Here we change it to a clean version by the do-copy-undo trick, and also replace its adders and

multiqubit Toffoli gates with the more efficient ones of [52] and [50] respectively. This leads to a circuit that contains

$$T_{\text{count}}(\text{PPoly}_{n,p,M,d}) = 4MT_{\text{count}}(\text{COMP_CONST}_n) + 2d[T_{\text{count}}(\text{MUL}_{n,p}) + T_{\text{count}}(\text{ADD}_n)] + 4dMT_{\text{count}}(\text{Toffoli}_{\lceil \log_2(M) \rceil + 1}) \quad (150)$$

$$= 4M(8n - 16) + 2d[(4n^2 - 8n + 8pn - 8p^2 + 8p) + (4n - 4)] + 16dM(\lceil \log_2(M) \rceil - 1) \quad (151)$$

$$= 8d(n^2 - n + 2pn - 2p^2 + 2p - 1) + 32M(n - 2) + 16dM(\lceil \log_2(M) \rceil - 1) \quad (152)$$

T gates. Moreover, it has T-depth $4d \max(n^2 - 2n + 2pn - 2p^2 + 2p, M(\lceil \log_2(M) \rceil - 1)) + 16M(n - 2) + 4d(n - 1)$ and requires $(d + 4)n + 2\lceil \log_2(M) \rceil$ ancilla qubits.

Exponential. In our algorithms, we need to compute the exponentials of log returns which are then used to compute the payoff of an option. We adopt the strategy of [54] to approximately implement such exponentiation, obtaining the unitary operation

$$\text{EXP}_{n,p,\epsilon} : |x\rangle |0\rangle \rightarrow |x\rangle \left| \overline{\text{pp}_{\text{exp}}(x)} \right\rangle, \quad (153)$$

for $x \in S_{n,p} \cap D$ such that $|e^x| < 2^{p-1}$. Here $\epsilon > 0$ is the target accuracy, and pp_{exp} is an (M_ϵ, d_ϵ) -piecewise polynomial that is ϵ -close to $\exp(x)$ on the relevant domain D . In our case, the domain is $D = [R_{\min}, R_{\max}]$, where $R_{\min}$ and $R_{\max}$ are the minimum and maximum possible log returns at any time in the stochastic process, respectively. The cost of implementing $\text{EXP}_{n,p,\epsilon}$ is identical to that of $\text{PPoly}_{n,p,M_\epsilon,d_\epsilon}$.

Arcsin of Square Root. The implementation of U_3 in Section 4.4 requires to compute the arcsin of square root of a given non-negative number, i.e., $\arcsin(\sqrt{x})$ for $x \geq 0$. We employ the strategy of [6] (which generalizes the one of [54]) to approximately implement this arithmetic operation, obtaining the unitary operation

$$\text{ARCSIN_SQRT}_{n,p,\epsilon} : |x\rangle |0\rangle \rightarrow |x\rangle \left| \overline{g(x)} \right\rangle. \quad (154)$$

for all $x \in S_{n,p} \cap [0, 1]$, where

$$g(x) = \begin{cases} \text{pp}_{\arcsin}(\sqrt{x}), & \text{if } x < 1/4, \\ \frac{\pi}{2} - \text{pp}_{\arcsin}(\sqrt{1-x}), & \text{otherwise,} \end{cases} \quad (155)$$

in which $\text{pp}_{\arcsin}$ is an (M_ϵ, d_ϵ) -piecewise polynomial that is ϵ -close to $\arcsin(x)$ on the domain $[-1/2, 1/2]$. Note that

$$\arcsin(\sqrt{x}) = \frac{\pi}{2} - \arcsin(\sqrt{1-x}), \quad (156)$$

for all $x \in [0, 1]$. Thus, if we ignore the round-off error in square rooting, then $g(x)$ is always ϵ -close to $\arcsin(\sqrt{x})$ regardless of the value of x .

Specifically, given an initial state $|x\rangle |0\rangle$, we run the following steps to obtain the target state $|x\rangle \left| \overline{g(x)} \right\rangle$:

1. Append a 1-qubit register W and set its state to $|z\rangle$, where $z = 1$ if $x < 1/4$, and 0 otherwise, by using $\text{COMP_CONST}_n(1/4)$.

2. Append an n -qubit register X and set its state to $|y\rangle$, where $y = x$ if $z = 1$, and $1 - x$ otherwise, by using $c - \text{SUB}_n$ and Clifford gates.
3. Append an n -qubit register Y and set its state to $|w\rangle$, where $w = \sqrt{y}$, by using SQRT_n .
4. Append an n -qubit register Z and set its state to $|v\rangle$, where $v = \overline{\text{pparcsin}(w)}$, by using $\text{PPoly}_{n,p,M_\epsilon,d_\epsilon}$.
5. Set the state of register Z to $|v\rangle$ if $z = 1$, and $|\pi/2 - v\rangle$ otherwise, by using $c - \text{SUB}_n$ and Clifford gates.
6. Copy the value of register Z to the target register by using CNOT gates.
7. Perform the inverse of the first five steps to restore the ancilla registers W , X , Y and Z to the zero state.

The resulting circuit has T-count

$$\begin{aligned} T_{\text{count}}(\text{ARCSIN_SQRT}_{n,p,\epsilon}) &= 2T_{\text{count}}(\text{PPoly}_{n,p,M_\epsilon,d_\epsilon}) + 2T_{\text{count}}(\text{COMP_CONST}_n) \\ &\quad + 2T_{\text{count}}(\text{SQRT}_n) + 4T_{\text{count}}(c - \text{SUB}_n) \end{aligned} \quad (157)$$

$$\begin{aligned} &= 16d_\epsilon(n^2 - n + 2pn - 2p^2 + 2p - 1) + 64M_\epsilon(n - 2) \\ &\quad + 32d_\epsilon M_\epsilon(\lceil \log_2(M_\epsilon) \rceil - 1) + 16\lceil n/2 \rceil^2 + 48n \\ &\quad + 64\lceil n/2 \rceil - 64 \end{aligned} \quad (158)$$

Similarly, one can get that its T-depth is $8d_\epsilon \max(n^2 - 2n + 2pn - 2p^2 + 2p, M_\epsilon(\lceil \log_2(M_\epsilon) \rceil - 1)) + 32M_\epsilon(n - 2) + 8d_\epsilon(n - 1) + 8\lceil n/2 \rceil^2 + 24n + 32\lceil n/2 \rceil - 32$. In addition, the circuit requires $(d_\epsilon + 7)n + 2\lceil \log_2(M_\epsilon) \rceil + 1$ ancilla qubits.

We summarize the costs of the above quantum arithmetic operations in Table 8.

B Resource-optimized block-encoding of the sine function

In this appendix, we present a low-cost block-encoding of the sine function. This construction is useful for the implementation of $U_{3,2}$ in Section 4.4 and U_{gauss} in Appendix C.

Let $n \in \mathbb{Z}^+$, $N = 2^n$, and $a, b \in \mathbb{R}$ satisfy $-1 \leq a < b \leq 1$. We aim to implement an $(n + 1)$ -qubit unitary operation U_{sin} such that

$$U_{\text{sin}} |i\rangle |0\rangle = |i\rangle [\sin(y(i)) |0\rangle + \cos(y(i)) |1\rangle], \quad \forall i = 0, 1, \dots, N - 1, \quad (159)$$

where $y(i) = a + (b - a)i/N$ for each i . Note that U_{sin} satisfies

$$(I_n \otimes \langle 0|) U_{\text{sin}} (I_n \otimes |0\rangle) = \sum_{i=0}^{N-1} \sin(y(i)) |i\rangle \langle i|. \quad (160)$$

In other words, U_{sin} is a block-encoding of the (shifted) sine function.

We assume the Clifford + T gate set, and aim to optimize the T-count and depth, while neglecting the costs of Clifford gates. This is so because the quantum computational costs are dominated by T gates [23]. In the circuits that implement the state-preparation method in [24], many of the T gates are used in the fault-tolerant implementation of

Operation	T-count	T-depth	Number of ancilla qubits	Techniques
Toffoli _n	4n − 8	n − 2	n − 1	[50]
Toffoli _n	16n − 60	16n − 60	1	[51]
ADD _n	4n − 4	2n − 2	n − 1	[52]
SUB _n	4n − 4	2n − 2	n − 1	[52]
c − ADD _n	8n − 4	4n − 2	2n − 1	[52]
c − SUB _n	8n − 4	4n − 2	2n − 1	[52]
ADD_CONST _n	4n − 8	2n − 4	2n − 2	[52]
SUB_CONST _n	4n − 8	2n − 4	2n − 2	[52]
COMP_CONST _n	8n − 16	4n − 8	3n − 2	[52]
MUL _{n,p}	4n ² − 8n + 8pn − 8p ² + 8p	2n ² − 4n + 4pn − 4p ² + 4p	2n − 1	[52]
MUL_CONST _{n,p}	2n ² − 6n + 4pn − 4p ² + 4p	n ² − 3n + 2pn − 2p ² + 2p	n − 1	[52]
SQRT _n	8⌈n/2⌉ ² + 32⌈n/2⌉ − 8	4⌈n/2⌉ ² + 16⌈n/2⌉ − 4	⌈3.5n⌉	[52], [53]
PPOLY _{n,p,M,d}	8d(n ² − n + 2pn − 2p ² + 2p − 1) + 32M(n − 2) + 16dM(⌈log ₂ (M)⌉ − 1)	4d max(n ² − 2n + 2pn − 2p ² + 2p, M(⌈log ₂ (M)⌉ − 1)) + 16M(n − 2) + 4d(n − 1)	(d + 4)n + 2⌈log ₂ (M)⌉	[50], [52], [54]
EXP _{n,p,ε}	8d _ε (n ² − n + 2pn − 2p ² + 2p − 1) + 32M _ε (n − 2) + 16d _ε M _ε (⌈log ₂ (M _ε)⌉ − 1)	4d _ε max(n ² − 2n + 2pn − 2p ² + 2p, M _ε (⌈log ₂ (M _ε)⌉ − 1)) + 16M _ε (n − 2) + 4d _ε (n − 1)	(d _ε + 4)n + 2⌈log ₂ (M _ε)⌉	[50], [52], [54]
ARCSIN_SQRT _{n,p,ε}	16d _ε (n ² − n + 2pn − 2p ² + 2p − 1) + 64M _ε (n − 2) + 32d _ε M _ε (⌈log ₂ (M _ε)⌉ − 1) + 16⌈n/2⌉ ² + 48n + 64⌈n/2⌉ − 64	8d _ε max(n ² − 2n + 2pn − 2p ² + 2p, M _ε (⌈log ₂ (M _ε)⌉ − 1)) + 32M _ε (n − 2) + 8d _ε (n − 1) + 8⌈n/2⌉ ² + 24n + 32⌈n/2⌉ − 32	(d _ε + 7)n + 2⌈log ₂ (M _ε)⌉ + 1	[6], [50], [52], [54]

Table 8: The costs of implementing the fixed-point quantum arithmetic operations used in this work.

where the R_z -count and R_z -depth have been reduced to $n + 1$ and 1, respectively, at the cost of n reusable ancilla qubits. Furthermore, we note that this circuit optimization is applicable to any series of consecutive controlled-rotation gates about the same axis, which are applied to the same target qubit and controlled by different qubits, e.g., (161), regardless of the rotation angles.

Next, we show an optimized implementation of a layer of n R_z gates of the form $\otimes_{i=0}^{n-1} R_z(2^i \theta)$, which we use to apply the gates in the dashed box in (165). We begin with the phase catalysis circuit from [56]:

$$(166)$$

where we have borrowed the notation for a relative-phase Toffoli gate, also known as temporary logical-AND, and its measurement-feedforward uncomputation, i.e.,

$$(167)$$

from [52]. Using (166), two $R_z(\theta)$ can be applied at the cost of a $R_z(2\theta)$ and a $R_z(\theta)|+\rangle$ catalyst state. We now show that the desired layer of R_z gates can be effected by applying (166) recursively. In particular, instead of applying the $R_z(2\theta)$ in (166) directly, we apply it using another (166) circuit where we let θ be replaced by 2θ ; in the second application of (166), the $R_z(4\theta)$ is effected using yet another (166), and so on and so forth. In order to apply $\otimes_{i=0}^{n-1} R_z(2^i \theta)$, (166) needs to be applied n times recursively, as shown below

$$(168)$$

We apply this recursive phase catalysis circuit with $\theta = -2^{-n} \Delta$, and remove $|\psi_0\rangle$ and the CNOTs controlled by it to implement the desired R_z layer, i.e., $\otimes_{i=1}^n R_z(-2^{-i} \Delta)$, in (165). It is worth pointing out that this circuit is directly applicable to quantum simulation algorithms [57, 58, 59, 60, 61] where R_z layers of this exact form are widely used to perform phase kickbacks.

Now we analyze the savings in computational costs due to our optimization. Here, we synthesize each R_z rotation using repeat-until-success (RUS) circuits in [37]. Then, each R_z rotation costs $1.15 \log_2(1/\epsilon)$ T gates, where ϵ is the synthesis error per rotation gate.

Suppose we aim to implement a $U_{\sin}$ with precision $\epsilon_{\sin}$. Then, each R_z rotation in the unoptimized circuit, i.e., (161) compiled using (162), can incur at most $\epsilon_{\sin}/(2n)$ error, and thus costs $1.15 \log_2(2n/\epsilon_{\sin})$ T gates. Then, the entire circuit costs $3.3n \log_2(2n/\epsilon_{\sin})$ T gates. The T-depth is the same as the T-count. Moreover, the circuit requires 1 ancilla qubit (which is necessary for the synthesis of all R_z rotations).

In contrast, our optimized circuit, assisted by a catalyst state, i.e., $\bigotimes_{i=0}^{n-1} R_z(2^i \theta) |+\rangle^{\otimes n}$, requires only n relative-phase Toffoli gates and two R_z gates. Since the catalyst state only needs to be synthesized once, after which it can be reused, and $U_{\sin}$ is typically invoked many times, i.e., $\gg n$, in an algorithm, for the sake of comparing with the unoptimized circuit, we neglect the one-time synthesis cost of $|\psi_g\rangle$, which we will address shortly. Then, each R_z has an error budget of $\epsilon_{\sin}/2$, and thus costs $1.15 \log_2(2/\epsilon_{\sin})$ T gates. In total, the circuit costs $4n + 3.3 \log_2(2/\epsilon_{\sin})$ T gates. The T-depth is then $n + 1.15 \log_2(2/\epsilon_{\sin}) + 1$. The number of ancilla qubits required is $2n + 2$ (two for synthesizing the R_z gates simultaneously and the remaining $3n$ are shown in (168)). We stress that the ancilla qubit count is insignificant compared to that required by other subroutines, e.g., arithmetic circuits, of the overall pricing algorithm. In summary, our optimization reduces the T-count and T-depth of $U_{\sin}$ from $O(n \log_2(n/\epsilon_{\sin}))$ to $O(n + \log_2(1/\epsilon_{\sin}))$, while retaining small constant factors and effectively not affecting the overall qubit count.

Let $U_{\sin}(n, \epsilon_{\sin})$ denote the final circuit consisting of Clifford and T gates. Our results can be summarized as follows. With the conventional method, we have

$$T_{\text{count}}(U_{\sin}(n, \epsilon_{\sin})) = 3.3n \log_2(2n/\epsilon_{\sin}), \quad (169)$$

$$T_{\text{depth}}(U_{\sin}(n, \epsilon_{\sin})) = 3.3n \log_2(2n/\epsilon_{\sin}), \quad (170)$$

$$\#\text{ancilla}(U_{\sin}(n, \epsilon_{\sin})) = 1, \quad (171)$$

whereas with our optimization, we get

$$T_{\text{count}}(U_{\sin}(n, \epsilon_{\sin})) = 4n + 3.3 \log_2(2/\epsilon_{\sin}), \quad (172)$$

$$T_{\text{depth}}(U_{\sin}(n, \epsilon_{\sin})) = n + 1.15 \log_2(2/\epsilon_{\sin}) + 1, \quad (173)$$

$$\#\text{ancilla}(U_{\sin}(n, \epsilon_{\sin})) = 3n + 2. \quad (174)$$

Returning to the catalyst state synthesis, we note that the state can be synthesized with n R_z rotations at a cost of $1.15n \log_2(n/\epsilon_{\text{cat}})$, where ϵ_{cat} is the error budget for this synthesis. Note that this only needs to be done once, as the catalyst state can be reused. In our algorithm, we choose sufficiently small ϵ_{cat} so that the error in the final result caused by the imperfection of catalyst state preparation is negligible. For example, for the instances considered in Section 4.7, we can pick $\epsilon_{\text{cat}} = 10^{-50}$ to ensure that this error is at most 10^{-30} . Then the synthesis of the catalyst state consumes at most 10^5 T gates, whereas the other components of the algorithm take more than 10^{10} T gates. In other words, the cost of synthesizing the catalyst state is insignificant compared to the total cost of the algorithm.

C Quantum circuits for preparing the Gaussian states

Preparing a quantum state whose amplitudes are described by a given function is a fundamental problem in quantum information science and has been extensively studied in the

past decades (e.g. [62, 63, 64, 65, 66, 67, 68, 69]). In this appendix, we utilize the method of [24] to prepare quantum states that encode Gaussian distributions, as these states are necessary for the algorithm based on strong Euler scheme. This method has the merits that it does not rely on amplitude oracles (which are often implemented with expensive coherent arithmetic), requires few ancilla qubits, and has rigorous performance guarantee,

Let us first review the main result of [24]. Suppose we want to prepare an n -qubit state whose amplitudes are described by a function $f : [a, b] \rightarrow \mathbb{C}$, i.e.,

$$|\psi_f\rangle = \frac{1}{\mathcal{N}_f} \sum_{i=0}^{N-1} f(x(i)) |i\rangle, \quad (175)$$

where $N = 2^n$, $x(i) = a + (b - a)i/N$ for each i , and $\mathcal{N}_f = \sqrt{\sum_{i=0}^{N-1} |f(x(i))|^2}$. Let

$$\|f\|_2^{[\infty]} = \sqrt{\int_a^b |f(x)|^2 dx}, \quad (176)$$

$$\|f\|_2^{[N]} = \sqrt{\frac{b-a}{N} \sum_{i=0}^{N-1} |f(x(i))|^2}, \quad (177)$$

be the L^2 norm of f and its discrete approximation. Then let

$$F_f^{[\infty]} = \frac{\|f\|_2^{[\infty]}}{\sqrt{(b-a)|f|_{\max}^2}}, \quad (178)$$

$$F_f^{[N]} = \frac{\|f\|_2^{[N]}}{\sqrt{(b-a)|f|_{\max}^2}}. \quad (179)$$

be the L^2 -norm filling-fraction of f and its discrete approximation. Theorem 1 of [24] states that if there exists a polynomial p of degree d_δ such that

$$\left| p(\sin(j/N)) - \frac{f(x(j))}{|f|_{\max}} \right| \leq \delta = \epsilon \cdot \min(F_f^{[N]}, F_{\tilde{f}}^{[N]}), \quad (180)$$

for all $j \in [0, N]$, where $\tilde{f}(x(j)) = p(\sin(j/N))$, then we can prepare a quantum state $|\psi_{\tilde{f}}\rangle$ that is ϵ -close in trace distance to $|\psi_f\rangle$ using $O(nd_\delta/F_{\tilde{f}}^{[N]})$ gates.

Now we apply this method to prepare an n -qubit state whose amplitudes are described by a Gaussian function. Precisely, we have $f(x) = e^{-x^2/4}$, $a = -\eta$ and $b = \eta$ for appropriately chosen η in our case, and hence our target state is

$$|\psi\rangle = \frac{1}{Z} \sum_{i=0}^{N-1} e^{-\frac{x(i)^2}{4}} |i\rangle, \quad (181)$$

where $x(i) = (2i - N)\eta/N$ for $i = 0, 1, \dots, N-1$, and $Z = \sum_{i=0}^{N-1} e^{-x(i)^2/4}$. We follow the procedure of [24] to prepare this state within ϵ_{prep} precision in trace distance. This leads to a circuit consisting of multi-qubit Toffoli gates, controlled and uncontrolled $R_x/R_y/R_z$ rotations, and Clifford gates:

1. We combine circuits (165) and (168) (see Appendix B for details) to implement an $(n+1)$ -qubit unitary operation U_{sin} (shown in (161)) such that

$$(\langle 0| \otimes I_n) U_{\text{sin}} (|0\rangle \otimes I_n) = \sum_{i=0}^{N-1} \sin(x(i)/\eta) |i\rangle \langle i|. \quad (182)$$

The entire circuit requires $3n + 1$ ancilla qubits (relative to the n -qubit input, and n ancilla qubits are used to store a reusable catalyst state), $4n$ T gates (contributed by n relative-phase Toffoli gates), two R_z rotations and Clifford gates. Note that the $4n$ T gates can be arranged in $n + 1$ layers, and the two R_z rotations can be executed in parallel with the assistance of 2 extra ancilla qubits. See Appendix B for a discussion on the one-time synthesis cost of the catalyst, which we will neglect here.

2. Let $g(z) = \exp\left(-\frac{\eta^2}{4} \arcsin^2(z)\right)$ for $z \in [-\sin(1), \sin(1)]$. Then for any $x \in [-\eta, \eta]$, we have $f(x) = g(\sin(x/\eta))$. Corollary 1 in Appendix D of [24] shows that for any $\beta, \delta > 0$, there exists a polynomial p of degree at most

$$d_\delta = \frac{\frac{\pi^2}{8}\beta + \ln(1/\delta)}{1 - \sin(1)} - 1 \quad (183)$$

such that

$$\left| \exp\left(-\frac{\beta}{2} \arcsin^2(z)\right) - p(z) \right| \leq \delta, \quad (184)$$

for all $z \in [-\sin(1), \sin(1)]$. Here we set $\beta = \eta^2/2$ and $\delta = \epsilon_{\text{prep}} \cdot \min(F_f^{[N]}, F_{\tilde{f}}^{[N]})$. Note that the proof of Lemma 5 in Appendix E of [24] shows that $F_{\tilde{f}}^{[N]} \geq \frac{1}{5\sqrt[4]{\beta}}$ provided that $N \geq \sqrt{\beta}$ and $|f(x(i)) - \tilde{f}(x(i))| \leq 1/4$ for each i . This implies that $F_f^{[N]}, F_{\tilde{f}}^{[N]} \geq \sqrt[4]{2}/(5\sqrt{\eta})$ and $\delta \geq \epsilon_{\text{prep}} \sqrt[4]{2}/(5\sqrt{\eta})$ in our case.

Now let $\tilde{f}(x) = p(\sin(x/\eta))$. Then we have $|f(x) - \tilde{f}(x)| \leq \delta$ for all $x \in [-\eta, \eta]$. Moreover, we can use quantum eigenvalue transformation (QET) to obtain an $(n+3)$ -qubit unitary operation $U_{\tilde{f}}$ such that

$$(\langle 0|^{\otimes 3} \otimes I_n) U_{\tilde{f}} (|0\rangle^{\otimes 3} \otimes I_n) = \frac{1}{2} \sum_{i=0}^{N-1} \frac{\tilde{f}(x(i))}{|\tilde{f}|_{\max}} |i\rangle \langle i|. \quad (185)$$

This QET circuit requires 2 ancilla qubits (relative to U_{sin}), d_δ applications of $U_{\text{sin}}/U_{\text{sin}}^\dagger$, $2d_\delta$ controlled- R_z rotations, and Clifford gates.

3. Applying $U_{\tilde{f}} (I_3 \otimes H^{\otimes n})$ on $|0\rangle^{\otimes n+3}$ yields a state from which $|\psi_{\tilde{f}}\rangle$ can be obtained probabilistically:

$$(\langle 0|^{\otimes 3} \otimes I_n) U_{\tilde{f}} (I_3 \otimes H^{\otimes n}) |0\rangle^{\otimes n+3} = \frac{F_{\tilde{f}}^{[N]}}{2} |\psi_{\tilde{f}}\rangle. \quad (186)$$

One can use exact amplitude amplification (EAA) [24] to boost this probability to one. This requires one ancilla qubit (relative to $U_{\tilde{f}}$), $2k + 1$ applications of $U_{\tilde{f}}/U_{\tilde{f}}^\dagger$, $2k + 1$ R_y rotations, k 4-qubit Toffoli gates, k $(n+4)$ -qubit Toffoli gates, and Clifford gates, where

$$k = \left\lceil \frac{\pi}{4 \arcsin\left(F_{\tilde{f}}^{[N]}/2\right)} - \frac{1}{2} \right\rceil. \quad (187)$$

Overall, the above circuit acts on $4n + 4$ qubits and contains

- $4nd_\delta(2k+1)$ T gates,
- $2d_\delta(2k+1)$ R_z rotations,
- $2d_\delta(2k+1)$ controlled- R_z rotations,
- $2k+1$ R_y rotations,
- k $(n+4)$ -qubit Toffoli gates,
- k 4-qubit Toffoli gates,

and Clifford gates. Note that among the $4n+4$ qubits, n of them are used to store a reusable catalyst state. Moreover, the $4nd_\delta(2k+1)$ T gates can be arranged in $d_\delta(n+1)(2k+1)$ layers, and the $2d_\delta(2k+1)$ R_z rotations can be executed as $d_\delta(2k+1)$ layers with the help of 2 additional ancilla qubits.

Next, we use the method of [50] to decompose the multi-qubit Toffoli gates into Clifford and T gates. Meanwhile, we also decompose each controlled- R_z rotation into two R_z rotations and Clifford gates as implied by (162). Consequently, the total number of R_y/R_z rotations in the circuit becomes

$$M_\delta = (6d_\delta + 1)(2k + 1). \quad (188)$$

Suppose we want to implement the whole circuit with ϵ_{gauss} precision. Then each R_y/R_z rotation can incur at most $\epsilon_{\text{gauss}}/M_\delta$ error. We use the method of [37] to synthesize each R_y/R_z rotation with this precision, which requires $1.15 \log_2(M_\delta/\epsilon_{\text{gauss}})$ T gates and Clifford gates with the help of an additional ancilla qubit.

Let $U_{\text{gauss}}(n, \eta, \epsilon_{\text{prep}}, \epsilon_{\text{gauss}})$ denote the final circuit consisting of Clifford and T gates. We conclude that

$$\begin{aligned} T_{\text{count}}(U_{\text{gauss}}(n, \eta, \epsilon_{\text{prep}}, \epsilon_{\text{gauss}})) &= 4nd_\delta(2k+1) + kT_{\text{count}}(\text{Toffoli}_{n+4}) + kT_{\text{count}}(\text{Toffoli}_4) \\ &\quad + M_\delta \cdot 1.15 \log_2(M_\delta/\epsilon_{\text{gauss}}) \\ &= 4nd_\delta(2k+1) + 4k(n+4) \\ &\quad + 1.15(6d_\delta + 1)(2k+1) \log_2(M_\delta/\epsilon_{\text{gauss}}), \end{aligned} \quad (189)$$

$$\begin{aligned} T_{\text{depth}}(U_{\text{gauss}}(n, \eta, \epsilon_{\text{prep}}, \epsilon_{\text{gauss}})) &= d_\delta(n+1)(2k+1) + kT_{\text{depth}}(\text{Toffoli}_{n+4}) + kT_{\text{depth}}(\text{Toffoli}_4) \\ &\quad + (5d_\delta + 1)(2k+1) \cdot 1.15 \log_2(M_\delta/\epsilon_{\text{gauss}}) \\ &= d_\delta(n+1)(2k+1) + k(n+4) \\ &\quad + 1.15(5d_\delta + 1)(2k+1) \log_2(M_\delta/\epsilon_{\text{gauss}}), \end{aligned} \quad (190)$$

$$\begin{aligned} \# \text{ancilla}(U_{\text{gauss}}(n, \eta, \epsilon_{\text{prep}}, \epsilon_{\text{gauss}})) &= 4 + n + \max(\# \text{ancilla}(\text{Toffoli}_{n+4}), 2n+2) \\ &= 3n+6. \end{aligned} \quad (191)$$

Note that the M_δ R_y/R_z rotations are arranged in $(5d_\delta + 1)(2k+1)$ layers instead of $(6d_\delta + 1)(2k+1)$ layers because, as mentioned above, we execute each pair of R_z rotations in the circuit for U_{sin} simultaneously with the assistance of 2 extra ancilla qubits. In addition, about the ancilla qubits, we need n of them to store a reusable catalyst state, 1 to construct U_{sin} , 2 for QET, and 1 for EAA. These $n+4$ ancilla qubits cannot be used to facilitate the implementation of R_y/R_z rotations and multiqubit Toffoli gates. For those operations, we need additional $2n+2$ ancilla qubits. Thus, the total number of ancilla qubits is $3n+6$ which is insignificant compared to that required by other components of the overall pricing algorithm.

References

- [1] Dylan Herman, Cody Googin, Xiaoyuan Liu, Yue Sun, Alexey Galda, Ilya Safro, Marco Pistoia, and Yuri Alexeev. “Quantum computing for finance”. *Nature Reviews Physics* **5**, 450–465 (2023).
- [2] Patrick Rebentrost, Brajesh Gupta, and Thomas R. Bromley. “Quantum computational finance: Monte carlo pricing of financial derivatives”. *Phys. Rev. A* **98**, 022321 (2018).
- [3] Nikitas Stamatopoulos, Daniel J. Egger, Yue Sun, Christa Zoufal, Raban Iten, Ning Shen, and Stefan Woerner. “Option Pricing using Quantum Computers”. *Quantum* **4**, 291 (2020).
- [4] Nikitas Stamatopoulos, Guglielmo Mazzola, Stefan Woerner, and William J. Zeng. “Towards Quantum Advantage in Financial Market Risk using Quantum Gradient Algorithms”. *Quantum* **6**, 770 (2022).
- [5] Dong An, Noah Linden, Jin-Peng Liu, Ashley Montanaro, Changpeng Shao, and Jiasu Wang. “Quantum-accelerated multilevel Monte Carlo methods for stochastic differential equations in mathematical finance”. *Quantum* **5**, 481 (2021).
- [6] Shouvanik Chakrabarti, Rajiv Krishnakumar, Guglielmo Mazzola, Nikitas Stamatopoulos, Stefan Woerner, and William J. Zeng. “A Threshold for Quantum Advantage in Derivative Pricing”. *Quantum* **5**, 463 (2021).
- [7] Javier Alcazar, Andrea Cadarso, Amara Katabarwa, Marta Mauri, Borja Peropadre, Guoming Wang, and Yudong Cao. “Quantum algorithm for credit valuation adjustments”. *New Journal of Physics* **24**, 023036 (2022).
- [8] Adam Bouland, Aditi Dandapani, and Anupam Prakash. “A quantum spectral method for simulating stochastic processes, with applications to monte carlo” (2023). [arXiv:2303.06719](https://arxiv.org/abs/2303.06719).
- [9] Nikitas Stamatopoulos and William J. Zeng. “Derivative Pricing using Quantum Signal Processing”. *Quantum* **8**, 1322 (2024).
- [10] Steven Shreve. “Stochastic Calculus for Finance II: Continuous-Time Models”. Volume 11. Springer New York, NY. (2004).
- [11] Steven Shreve. “Stochastic Calculus for Finance I: The Binomial Asset Pricing Model”. Springer New York, NY. (2005).
- [12] Peter E. Kloeden and Eckhard Platen. “Numerical Solution of Stochastic Differential Equations”. Springer Berlin, Heidelberg. (1992).
- [13] Gilles Brassard, Peter Høyer, Michele Mosca, and Alain Tapp. “Quantum amplitude amplification and estimation”. *Quantum Computation and Information* **305**, 53–74 (2002).
- [14] Fischer Black and Myron Scholes. “The pricing of options and corporate liabilities”. *Journal of Political Economy* **81**, 637–654 (1973).
- [15] Robert C Merton. “Theory of rational option pricing”. *The Bell Journal of economics and management science* **4**, 141–183 (1973).
- [16] Steven L Heston. “A closed-form solution for options with stochastic volatility with applications to bond and currency options”. *The Review of Financial Studies* **6**, 327–343 (1993).
- [17] Natalia A Beliaeva et al. “A simple approach to pricing american options under the heston stochastic volatility model”. *The Journal of Derivatives* **17**, 25–43 (2010).
- [18] Elisa Alòs. “A decomposition formula for option prices in the heston model and applications to option pricing approximation”. *Finance and Stochastics* **16**, 403–422 (2012).

- [19] Carl Chiarella, Boda Kang, and Gunter H. Meyer. “The evaluation of barrier option prices under stochastic volatility”. *Computers & Mathematics with Applications* **64**, 2034–2048 (2012).
- [20] Xin-Jiang He and Song-Ping Zhu. “A closed-form pricing formula for european options under the heston model with stochastic interest rate”. *Journal of Computational and Applied Mathematics* **335**, 323–333 (2018).
- [21] Dmitry Grinko, Julien Gacon, Christa Zoufal, and Stefan Woerner. “Iterative quantum amplitude estimation”. *npj Quantum Information* **7**, 52 (2021).
- [22] Michael A. Nielsen and Isaac L. Chuang. “Quantum Computation and Quantum Information”. *Cambridge University Press*. Cambridge, U.K. (2000).
- [23] Sergey Bravyi and Alexei Kitaev. “Universal quantum computation with ideal clifford gates and noisy ancillas”. *Phys. Rev. A* **71**, 022316 (2005).
- [24] Sam McArdle, András Gilyén, and Mario Berta. “Quantum state preparation without coherent arithmetic” (2022). [arXiv:2210.14892](https://arxiv.org/abs/2210.14892).
- [25] Desmond J. Higham. “An algorithmic introduction to numerical simulation of stochastic differential equations”. *SIAM Review* **43**, 525–546 (2001).
- [26] Leif BG Andersen and Rupert Brotherton-Ratcliffe. “Extended libor market models with stochastic volatility”. [Available at SSRN 294853](https://ssrn.com/abstract=294853) (2001).
- [27] Paul Glasserman. “Monte Carlo Methods in Financial Engineering”. *Springer New York, NY*. (2003).
- [28] Grigori N Milstein and Michael V Tretyakov. “Stochastic numerics for mathematical physics”. *Springer Berlin, Heidelberg*. (2004).
- [29] Michael B Giles. “Multilevel monte carlo path simulation”. *Operations Research* **56**, 607–617 (2008).
- [30] Roger Lord, Remmert Koekkoek, and Dick Van Dijk. “A comparison of biased simulation schemes for stochastic volatility models”. *Quantitative Finance* **10**, 177–194 (2010).
- [31] Scott Aaronson and Patrick Rall. “Quantum approximate counting, simplified”. In *Symposium on Simplicity in Algorithms*. *Pages 24–32*. SIAM (2020).
- [32] Yohichi Suzuki, Shumpei Uno, Rudy Raymond, Tomoki Tanaka, Tamiya Onodera, and Naoki Yamamoto. “Amplitude estimation without phase estimation”. *Quantum Information Processing* **19**, 1–17 (2020).
- [33] Guoming Wang, Dax Enshan Koh, Peter D. Johnson, and Yudong Cao. “Minimizing estimation runtime on noisy quantum computers”. *PRX Quantum* **2**, 010346 (2021).
- [34] Tudor Giurgica-Tiron, Iordanis Kerenidis, Farrokh Labib, Anupam Prakash, and William Zeng. “Low depth algorithms for quantum amplitude estimation”. *Quantum* **6**, 745 (2022).
- [35] Kirill Plekhanov, Matthias Rosenkranz, Mattia Fiorentini, and Michael Lubasch. “Variational quantum amplitude estimation”. *Quantum* **6**, 670 (2022).
- [36] Aram W. Harrow, Avinandan Hassidim, and Seth Lloyd. “Quantum algorithm for linear systems of equations”. *Phys. Rev. Lett.* **103**, 150502 (2009).
- [37] Alex Bocharov, Martin Roetteler, and Krysta M. Svore. “Efficient synthesis of universal repeat-until-success quantum circuits”. *Phys. Rev. Lett.* **114**, 080502 (2015).
- [38] Joonho Lee, Dominic W. Berry, Craig Gidney, William J. Huggins, Jarrod R. McClean, Nathan Wiebe, and Ryan Babbush. “Even more efficient quantum computations of chemistry through tensor hypercontraction”. *PRX Quantum* **2**, 030305 (2021).
- [39] Nicholas C. Rubin, Dominic W. Berry, Fionn D. Malone, Alec F. White, Tanuj Khat-tar, A. Eugene DePrince, Sabrina Sicolo, Michael Küehn, Michael Kaicher, Joonho

- Lee, and Ryan Babbush. “Fault-tolerant quantum simulation of materials using bloch orbitals”. *PRX Quantum* **4**, 040303 (2023).
- [40] Vera von Burg, Guang Hao Low, Thomas Häner, Damian S. Steiger, Markus Reiher, Martin Roetteler, and Matthias Troyer. “Quantum computing enhanced computational catalysis”. *Phys. Rev. Res.* **3**, 033055 (2021).
- [41] Dominic W. Berry, Craig Gidney, Mario Motta, Jarrod R. McClean, and Ryan Babbush. “Qubitization of Arbitrary Basis Quantum Chemistry Leveraging Sparsity and Low Rank Factorization”. *Quantum* **3**, 208 (2019).
- [42] João F. Doriguello, Alessandro Luongo, Jinge Bao, Patrick Rebentrost, and Miklos Santha. “Quantum Algorithm for Stochastic Optimal Stopping Problems with Applications in Finance”. In François Le Gall and Tomoyuki Morimae, editors, 17th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2022). Volume 232 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 2:1–2:24. Dagstuhl, Germany (2022). Schloss Dagstuhl – Leibniz-Zentrum für Informatik.
- [43] Dax Enshan Koh, Guoming Wang, Peter D. Johnson, and Yudong Cao. “Foundations for Bayesian inference with engineered likelihood functions for robust amplitude estimation”. *Journal of Mathematical Physics* **63**, 052202 (2022).
- [44] Ruizhe Zhang, Guoming Wang, and Peter Johnson. “Computing Ground State Properties with Early Fault-Tolerant Quantum Computers”. *Quantum* **6**, 761 (2022).
- [45] Guoming Wang, Sukin Sim, and Peter D. Johnson. “State Preparation Boosters for Early Fault-Tolerant Quantum Computation”. *Quantum* **6**, 829 (2022).
- [46] Guoming Wang. “Classically-boosted quantum optimization algorithm” (2022). [arXiv:2203.13936](https://arxiv.org/abs/2203.13936).
- [47] Guoming Wang, Daniel Stilck França, Ruizhe Zhang, Shuchen Zhu, and Peter D. Johnson. “Quantum algorithm for ground state energy estimation using circuit depth with exponentially improved dependence on precision”. *Quantum* **7**, 1167 (2023).
- [48] Guoming Wang, Daniel Stilck França, Gumaro Rendon, and Peter D. Johnson. “Faster ground state energy estimation on early fault-tolerant quantum computers via rejection sampling” (2023). [arXiv:2304.09827](https://arxiv.org/abs/2304.09827).
- [49] Amara Katabarwa, Katerina Gratsea, Athena Caesura, and Peter D. Johnson. “Early fault-tolerant quantum computing”. *PRX Quantum* **5**, 020101 (2024).
- [50] Cody Jones. “Low-overhead constructions for the fault-tolerant toffoli gate”. *Phys. Rev. A* **87**, 022328 (2013).
- [51] Matthew Amy and Neil J. Ross. “Phase-state duality in reversible circuit design”. *Phys. Rev. A* **104**, 052602 (2021).
- [52] Craig Gidney. “Halving the cost of quantum addition”. *Quantum* **2**, 74 (2018).
- [53] Edgard Muñoz-Coreas and Himanshu Thapliyal. “T-count and qubit optimized quantum circuit design of the non-restoring square root algorithm”. *ACM Journal on Emerging Technologies in Computing Systems (JETC)* **14**, 1–15 (2018).
- [54] Thomas Häner, Martin Roetteler, and Krysta M Svore. “Optimizing quantum circuits for arithmetic” (2018). [arXiv:1805.12445](https://arxiv.org/abs/1805.12445).
- [55] Qingfeng Wang, Ming Li, Christopher Monroe, and Yunseong Nam. “Resource-Optimized Fermionic Local-Hamiltonian Simulation on a Quantum Computer for Quantum Chemistry”. *Quantum* **5**, 509 (2021).
- [56] Craig Gidney and Austin G. Fowler. “Efficient magic state factories with a catalyzed $|CCZ\rangle$ to $2|T\rangle$ transformation”. *Quantum* **3**, 135 (2019).
- [57] Yunseong Nam and Dmitri Maslov. “Low-cost quantum circuits for classically in-

- tractable instances of the hamiltonian dynamics simulation problem”. *npj Quantum Information* **5**, 44 (2019).
- [58] Alexander F. Shaw, Pavel Lougovski, Jesse R. Stryker, and Nathan Wiebe. “Quantum Algorithms for Simulating the Lattice Schwinger Model”. *Quantum* **4**, 306 (2020).
 - [59] Earl T Campbell. “Early fault-tolerant simulations of the hubbard model”. *Quantum Science and Technology* **7**, 015007 (2021).
 - [60] Angus Kan and Yunseong Nam. “Lattice quantum chromodynamics and electrodynamics on a universal quantum computer” (2022). [arXiv:2107.12769](#).
 - [61] Angus Kan and Yunseong Nam. “Simulating lattice quantum electrodynamics on a quantum computer”. *Quantum Science and Technology* **8**, 015008 (2022).
 - [62] Lov K. Grover. “Synthesis of quantum superpositions by quantum computation”. *Phys. Rev. Lett.* **85**, 1334–1337 (2000).
 - [63] Lov Grover and Terry Rudolph. “Creating superpositions that correspond to efficiently integrable probability distributions” (2002). [arXiv:quant-ph/0208112](#).
 - [64] Adam Holmes and A. Y. Matsuura. “Efficient quantum circuits for accurate state preparation of smooth, differentiable functions” (2020). [arXiv:2005.04351](#).
 - [65] Juan José García-Ripoll. “Quantum-inspired algorithms for multivariate analysis: from interpolation to partial differential equations”. *Quantum* **5**, 431 (2021).
 - [66] Sevag Gharibian, Zeph Landau, Seung Woo Shin, and Guoming Wang. “Tensor network non-zero testing”. *Quantum Info. Comput.* **15**, 885–889 (2015).
 - [67] Arthur G. Rattew and Bálint Koczor. “Preparing arbitrary continuous functions in quantum registers with logarithmic complexity” (2022). [arXiv:2205.00519](#).
 - [68] Johannes Bausch. “Fast Black-Box Quantum State Preparation”. *Quantum* **6**, 773 (2022).
 - [69] Shengbin Wang, Zhimin Wang, Runhong He, Shangshang Shi, Guolong Cui, Ruimin Shang, Jiayun Li, Yanan Li, Wendong Li, Zhiqiang Wei, and Yongjian Gu. “Inverse-coefficient black-box quantum state preparation”. *New Journal of Physics* **24**, 103004 (2022).