

Option Pricing using Quantum Computers

Nikitas Stamatopoulos,¹ Daniel J. Egger,² Yue Sun,¹ Christa Zoufal,^{2,3} Raban Iten,² Ning Shen,¹ and Stefan Woerner²

¹*J.P. Morgan Chase - Quantitative Research*

²*IBM Research - Zurich*

³*ETH Zurich*

(Dated: May 8, 2019)

We present a methodology to price options and portfolios of options on a gate-based quantum computer using amplitude estimation, an algorithm which provides a quadratic speedup compared to classical Monte Carlo methods. The options that we cover include vanilla options, multi-asset options and path-dependent options such as barrier options. We put an emphasis on the implementation of the quantum circuits required to build the input states and operators needed by amplitude estimation to price the different option types. Additionally, we show simulation results to highlight how the circuits that we implement price the different option contracts. Finally, we examine the performance of option pricing circuits on real quantum hardware using the IBM Q Tokyo quantum device.

I. INTRODUCTION

Options are financial derivative contracts that give the buyer the right, but not the obligation, to buy (call option) or sell (put option) an underlying asset at an agreed-upon price (strike) and timeframe (exercise window). In their simplest form, the strike price is a fixed value and the timeframe is a single point in time, but exotic variants may be defined on more than one underlying asset, the strike price can be a function of several market parameters and could allow for multiple exercise dates. As well as providing investors with a vehicle to profit by taking a view on the market or exploit arbitrage opportunities, options are core to various hedging strategies and as such, understanding their properties is a fundamental objective of financial engineering. For an overview of option types, features and uses, we refer the reader to Ref. [1].

Due to the stochastic nature of the parameters options are defined on, calculating their fair value can be an arduous task and while analytical models exist for the simplest types of options [2], the simplifying assumptions on the market dynamics required for the models to provide closed-form solutions often limit their applicability [3]. Hence, more often than not, numerical methods have to be employed for option pricing, with Monte Carlo being one of the most popular due to its flexibility and ability to generically handle stochastic parameters [4, 5]. However, despite their attractive features in option pricing, classical Monte Carlo methods generally require extensive computational resources to provide accurate option price estimates, particularly for complex options. Because of the widespread use of options in the finance industry, accelerating their convergence can have a significant impact in the operations of a financial institution.

By leveraging the laws of quantum mechanics a quantum computer [6] may provide novel ways to solve computationally intensive problems such as quantum chemistry [7–10], solving linear systems of equations [11], and machine learning [12–14]. Quantitative finance, a field with many computationally hard problems, may benefit from

quantum computing. Recently developed applications of gate-based quantum computing for use in finance [15] include portfolio optimization [16], the calculation of risk measures [17] and pricing derivatives [18–20]. Several of these applications are based on the Amplitude Estimation algorithm [21] which can estimate a parameter with a convergence rate of $1/M$, where M is the number of quantum samples used. This represents a theoretical quadratic speed-up compared to classical Monte Carlo methods.

In this paper we extend the pricing methodology presented in [17, 18] and place a strong emphasis on the implementation of the algorithms in a gate-based quantum computer. We first classify options according to their features and show how to take the different features into account in a quantum computing setting. In section III, we review the quantum algorithms needed to price options and discuss how to represent relevant probability distributions in a quantum computer. In section IV, we show a framework to price vanilla options and portfolios of vanilla options, options with path-dependent payoffs and options on several underlying assets. In section V we show results from quantum hardware.

II. REVIEW OF OPTION TYPES AND THEIR CHALLENGES

We classify options according to two categories: path-independent vs path-dependent and options on single assets or on multiple assets. Path-independent options have a payoff function that depends on an underlying asset at a single point in time. Therefore, the price of the asset up to the exercise date of the option is irrelevant for the option price. By contrast, the payoff of path-dependent options depends on the evolution of the price of the asset and its history up to the exercise date. Table I exemplifies this classification. Options that are path-independent and rely on a single asset are the easiest to price. This is done using Amplitude Estimation once a proper representation of the distribution of the underlying asset can

TABLE I. Example of the different option types.

	Single-asset	Multi-asset
Path-independent	European put/call	Basket option
Path-dependent	Barrier & asian options	Multi-asset barrier options

be loaded to the quantum computer. Path-independent options on multiple assets are only slightly harder to price since more than one asset is now involved and the probability distribution loaded into the quantum computer must account for correlations between the assets. Path-dependent options are harder to price than path-independent options since they require a representation of the possible paths the underlying assets can take in the quantum computer.

III. IMPLEMENTATION ON A GATE BASED QUANTUM COMPUTER

Here we review some of the building blocks needed to price options on a gate-based quantum computer.

A. Distribution loading

The analytical formulas used to price options in the Black-Scholes-Merton (BSM) model [2, 22] assume that the underlying stock prices at maturity follow a lognormal distribution with constant volatility. Such distributions can be efficiently loaded in a gate-based quantum computer [18, 23]. However, to properly model the market prices of options, the volatility of the geometric brownian process describing the evolution of the assets must be changed for options with different strike prices [24]. This discrepancy between the BSM model and market prices is because stocks do not follow a geometric Brownian motion process with constant volatility. It is thus important to be able to efficiently represent arbitrary distributions of financial data in a quantum computer.

The best known methods to load arbitrary states into quantum systems require exponentially many gates [25, 26], making it inefficient to model arbitrary distributions as quantum states. This limitation may be overcome by using quantum Generative Adversarial Networks (qGAN). These networks allow us to load a distribution using a polynomial number of gates [19]. A qGAN can learn the random distribution X underlying the observed data samples $\{x^0, \dots, x^{k-1}\}$ and load it directly into a quantum state. This generative model employs the interplay of a classical discriminator, a neural network [27], and a quantum generator (a parametrized quantum circuit). More specifically, the qGAN training consists of alternating optimization steps of the discriminator's parameters ϕ and the generator's parameters θ . After the

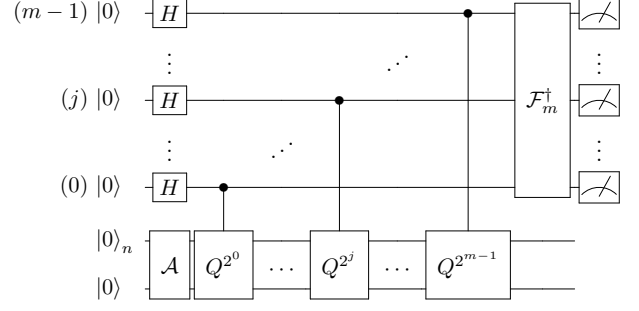


FIG. 1. The quantum circuit of amplitude estimation.

training, the output of the generator is a quantum state

$$|\psi(\theta)\rangle_n = \sum_{i=0}^{2^n-1} \sqrt{p_i(\theta)} |i\rangle_n, \quad (1)$$

that represents the target distribution. The n -qubit state $|i\rangle_n = |i_{n-1} \dots i_0\rangle$ encodes the integer $i = 2^{n-1}i_{n-1} + \dots + 2i_1 + i_0 \in \{0, \dots, 2^n - 1\}$ with $i_k \in \{0, 1\}$ and $k = 0, \dots, n-1$. The probabilities $p_i(\theta)$ approximate the random distribution underlying the training data. We note that the outcomes of a random variable X can be mapped to the integer set $\{0, \dots, 2^n - 1\}$ using an affine mapping.

B. Amplitude Estimation

The advantage of pricing options on a quantum computer comes from the amplitude estimation (AE) algorithm [21] which provides a quadratic speed-up over classical Monte-Carlo simulations [28, 29]. Suppose a unitary operator \mathcal{A} acting on a register of $(n+1)$ qubits such that

$$\mathcal{A}|0\rangle_{n+1} = \sqrt{1-a}|\psi_0\rangle_n|0\rangle + \sqrt{a}|\psi_1\rangle_n|1\rangle \quad (2)$$

for some normalized states $|\psi_0\rangle_n$ and $|\psi_1\rangle_n$, where $a \in [0, 1]$ is unknown. AE allows the efficient estimation of a , i.e., the probability of measuring $|1\rangle$ in the last qubit. This estimation is obtained with an operator Q , based on \mathcal{A} , and Quantum Phase Estimation [30] to approximate certain eigenvalues of Q . AE requires m additional sampling qubits and $M = 2^m$ applications of Q . The m qubits, initialized to an equal superposition state by Hadamard gates, are used to control different powers of Q . After applying an inverse Quantum Fourier Transform, their state is measured resulting in an integer $y \in \{0, \dots, M-1\}$, which is classically mapped to the estimator $\tilde{a} = \sin^2(y\pi/M) \in [0, 1]$, see the circuit in Fig. 1. The estimator \tilde{a} satisfies

$$|a - \tilde{a}| \leq \frac{\pi}{M} + \frac{\pi^2}{M^2} = \mathcal{O}(M^{-1}), \quad (3)$$

with probability of at least $8/\pi^2$. This represents a quadratic speedup compared to the $\mathcal{O}(M^{-1/2})$ convergence rate of classical Monte Carlo methods [31].

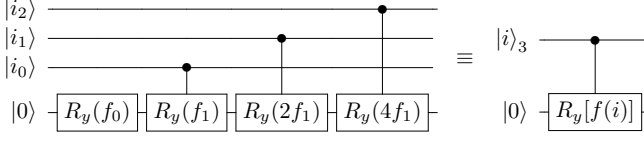


FIG. 2. Quantum circuit that creates the state in Eq. (4). Here, the independent variable $i = 4i_2 + 2i_1 + i_0 \in \{0, \dots, 7\}$ is encoded by three qubits in the state $|i\rangle_3 = |i_2 i_1 i_0\rangle$ with $i_k \in \{0, 1\}$. Therefore, the linear function $f(i) = f_1 i + f_0$ is given by $4f_1 i_2 + 2f_1 i_1 + f_1 i_0 + f_0$. After applying this circuit the quantum state is $|i\rangle_3 [\cos(f_1 i + f_0) |0\rangle + \sin(f_1 i + f_0) |1\rangle]$. The circuit on the left shows an abbreviated notation.

C. Linearly controlled Y-rotations

We obtain the expectation value of a linear function f of a random variable X with AE by creating the operator \mathcal{A} such that $a = \mathbb{E}[f(X)]$, see Eq. (2). Once \mathcal{A} is implemented we can prepare the state in Eq. (2) and the Q operator. In this section, we show how to create a close relative of the operator in Eq. (2) and then, in section IIID, we show how to use AE.

Since the payoff function for option portfolios is piecewise linear we only need to consider linear functions $f : \{0, \dots, 2^n - 1\} \rightarrow [0, 1]$ which we write $f(i) = f_1 i + f_0$. We can efficiently create an operator that performs

$$|i\rangle_n |0\rangle \rightarrow |i\rangle_n \{\cos[f(i)] |0\rangle + \sin[f(i)] |1\rangle\} \quad (4)$$

using controlled Y-rotations [17]. To implement the linear term of $f(i)$ each qubit j (where $j \in \{0, \dots, n-1\}$) in the $|i\rangle_n$ register acts as a control for a Y-rotation with angle $2^j f_1$ of the ancilla qubit. The constant term f_0 is implemented by a rotation of the ancilla qubit without any controls, see Fig. 2. The controlled Y-rotations can be implemented with CNOT and single-qubit gates [32].

D. Expectation value of functions using AE

We now describe how to obtain $\mathbb{E}[f(X)]$ for a linear function f of a random variable X which is mapped to integer values $i \in \{0, \dots, 2^n - 1\}$ that occur with probability p_i . To do this we create the operator that maps $\sum_i \sqrt{p_i} |i\rangle_n |0\rangle$ to

$$\sum_{i=0}^{2^n-1} \sqrt{p_i} |i\rangle_n \left[\cos\left(c\tilde{f}(i) + \frac{\pi}{4}\right) |0\rangle + \sin\left(c\tilde{f}(i) + \frac{\pi}{4}\right) |1\rangle \right]$$

using the procedure outlined in Sec. IIIC. The parameter $c \in [0, 1]$ is a scaling parameter. The functions $\tilde{f}(i)$ and $f(i)$ are related by

$$\tilde{f}(i) = 2 \frac{f(i) - f_{\min}}{f_{\max} - f_{\min}} - 1. \quad (5)$$

Here $f_{\min} = \min_i f(i)$ and $f_{\max} = \max_i f(i)$. The relation in Eq. (5) is chosen so that $\tilde{f}(i) \in [-1, 1]$. Thus,

$\sin^2[c\tilde{f}(i) + \pi/4]$ is an anti-symmetric function around $\tilde{f}(i) = 0$. With these definitions, the probability to find the ancilla qubit in state $|1\rangle$, namely

$$P_1 = \sum_{i=0}^{2^n-1} p_i \sin^2\left(c\tilde{f}(i) + \frac{\pi}{4}\right),$$

is well approximated by

$$\sum_{i=0}^{2^n-1} p_i \left(c\tilde{f}(i) + \frac{1}{2}\right) = c \frac{2\mathbb{E}[f(i)] - f_{\min}}{f_{\max} - f_{\min}} - c + \frac{1}{2}. \quad (6)$$

To obtain this result we made use of the approximation

$$\sin^2\left(c\tilde{f}(i) + \frac{\pi}{4}\right) = c\tilde{f}(i) + \frac{1}{2} + \mathcal{O}(c^3 \tilde{f}^3(i)) \quad (7)$$

which is valid for small values of $c\tilde{f}(i)$. With this first order approximation the convergence rate of AE is $\mathcal{O}(M^{-2/3})$ when c is properly chosen which is already faster than classical Monte Carlo methods [17]. We can recover the $\mathcal{O}(M^{-1})$ convergence rate of AE by using higher orders implemented with quantum arithmetic. The resulting circuits, however, have more gates. This trade-off, discussed in Ref. [17], also gives a formula that specifies which value of c to use to minimize the estimation error made when using AE. From Eq. (6) we can recover $\mathbb{E}[f(i)]$ since AE allows us to efficiently retrieve P_1 and because we know the values of f_{\min} , f_{\max} and c .

IV. OPTION PRICING ON A QUANTUM COMPUTER

In this section we show how to price the different options shown in Tab. I. We put an emphasis on the implementation of the quantum circuits that prepare the states needed by AE. We use the different building blocks reviewed in Sec. III.

A. Path-independent options

The price of path-independent *vanilla* options (e.g. European call and put options) depend only on the distribution of the underlying asset price S_T at the option maturity T and the payoff function $f(S_T)$ of the option. To encode the distribution of S_T in a quantum state we truncate it to the range $[S_{T,\min}, S_{T,\max}]$ and discretize this interval to $\{0, \dots, 2^n - 1\}$ using n qubits. In the quantum computer the distribution loading operator \mathcal{P}_X creates a state

$$|0\rangle_n \xrightarrow{\mathcal{P}_X} |\psi\rangle_n = \sum_{i=0}^{2^n-1} \sqrt{p_i} |i\rangle_n, \quad (8)$$

with $i \in \{0, \dots, 2^n - 1\}$ to represent S_T . This state, exemplified in Fig. 3, may be created using the methods discussed in Sec. IIIA.

We start by showing how to price vanilla call or put options and then generalize our method to capture the payoff structure of portfolios containing more than one vanilla option.

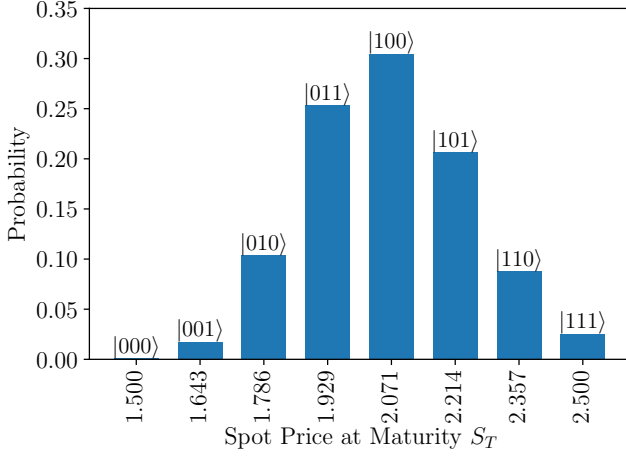


FIG. 3. Example price distribution at maturity loaded in a three-qubit register. In this example we followed the Black-Scholes-Merton model which implies a lognormal distribution of the asset price at maturity T with probability density function $P(S_T) = \frac{1}{S_T \sigma \sqrt{2\pi T}} \exp\left(-\frac{(\ln S_T - \mu)^2}{2\sigma^2 T}\right)$. σ is the volatility of the asset and $\mu = (r - 0.5\sigma^2)T + \ln(S_0)$, with r the risk-free market rate and S_0 the asset's spot at $t = 0$. In this figure we used $S_0 = 2$, $\sigma = 10\%$, $r = 4\%$ and $T = 300/365$.

1. Vanilla options

To price vanilla options with strike K , we implement a comparison between the values in state (8) with K . A quantum comparator circuit sets an ancilla qubit $|c\rangle$, initially in state $|0\rangle$, to the state $|1\rangle$ if $i \geq K$ and $|0\rangle$ otherwise. The state $|\psi\rangle_n$ in the quantum computer therefore undergoes the transformation

$$|\psi\rangle_n |0\rangle \rightarrow |\phi_1\rangle = \sum_{i < K} \sqrt{p_i} |i\rangle_n |0\rangle + \sum_{i \geq K} \sqrt{p_i} |i\rangle_n |1\rangle.$$

This operation can be implemented by a quantum comparator [33] based on CNOT and Toffoli gates. Since we know the value of the strike, we can implement a circuit tailored to the specific strike price. We use n ancilla qubits $|a_1, \dots, a_n\rangle$ and compute the two's complement of the strike price K in binary using n bits, storing the digits in a (classical) array $t[n]$. For each qubit $|i_k\rangle$ in the $|i\rangle_n$ register, with $k \in \{0, \dots, n-1\}$, we compute the possible carry bit of the bitwise addition of $|i_k\rangle$ and $t[k]$ into $|a_k\rangle$. If $t[k] = 0$, there is a carry qubit at position k only if there is a carry at position $k-1$ and $|i_k\rangle = 1$. If $t[k] = 1$, there is a carry qubit at position k if there is a carry at position $k-1$ or $|i_k\rangle = 1$. After going through all n qubits from least to most significant, $|i\rangle_n$ will be greater

or equal to the strike price, only if there is a carry at the last (most significant) qubit. This procedure along with the necessary gate operations is illustrated in Fig. 4. An implementation for $K = 1.9$ and a three-qubit register is shown in Fig. 6.

To prepare the operator for use with AE we add to $|\phi_1\rangle$ a second ancilla qubit initially in the state $\cos(g_0)|0\rangle + \sin(g_0)|1\rangle$. Here, g_0 is an angle with a value that we will carefully select. Next, we perform a rotation of the new ancilla qubit controlled by the comparator qubit $|c\rangle$ and the qubits in $|\psi\rangle_n$. The state $|\phi_1\rangle [\cos(g_0)|0\rangle + \sin(g_0)|1\rangle]$ is mapped to

$$\sum_{i < K} \sqrt{p_i} |i\rangle_n |0\rangle [\cos(g_0)|0\rangle + \sin(g_0)|1\rangle] + \sum_{i \geq K} \sqrt{p_i} |i\rangle_n |1\rangle \{\cos[g_0 + g(i)]|0\rangle + \sin[g_0 + g(i)]|1\rangle\}. \quad (9)$$

This operation, implemented by the quantum circuit in Fig. 7, applies a rotation with an angle $g(i)$ only if $i \geq K$. The probability to find the second ancilla in state $|1\rangle$, efficiently measurable using AE, is

$$P_1 = \sum_{i < K} p_i \sin^2(g_0) + \sum_{i \geq K} p_i \sin^2[g_0 + g(i)]. \quad (10)$$

Now, we must carefully choose the angle g_0 and the function $g(i)$ to recover the expected payoff $\mathbb{E}[f(i)]$ of the option from P_1 using the approximation in Eq. (6). We will further require that $g(i) \in [-c, c]$ for $i \geq K$ to maximize the accuracy. The payoff function of vanilla options is piece-wise linear

$$f(i) = \begin{cases} a_{<} \cdot i + b_{<} & i < K, \\ a_{\geq} \cdot i + b_{\geq} & i \geq K. \end{cases} \quad (11)$$

We now focus on a European call option with payoff $f(i) = \max(0, i - K)$, i.e., $a_{<} = b_{<} = 0$, $a_{\geq} = 1$, and $b_{\geq} = -K$. To reproduce $f(i) = i - K$ for $i \geq K$ and simultaneously satisfy $g_0 + g(i) - \pi/4 \in [-c, c]$, see Sec. III D, we must set

$$g(i) = \frac{2c(i - K)}{i_{\max} - K}, \quad (12)$$

where $i_{\max} = 2^n - 1$. This choice of $g(i)$ forces us to choose

$$g_0 = \frac{\pi}{4} - c. \quad (13)$$

To see why, we substitute Eqs. (12) and (13) in Eq. (10) and use the approximation in Eq. (7). Therefore,

$$\begin{aligned} P_1 &\approx \sum_{i < K} p_i \left(\frac{1}{2} - c\right) + \sum_{i \geq K} p_i \left(\frac{2c(i - K)}{i_{\max} - K} + \frac{1}{2} - c\right) \\ &= \frac{1}{2} - c + \frac{2c}{i_{\max} - K} \sum_{i \geq K} p_i (i - K). \end{aligned} \quad (14)$$

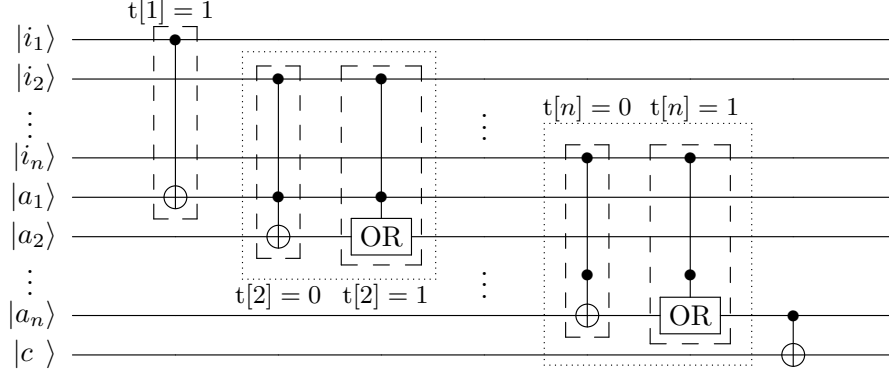


FIG. 4. Circuit that compares the value represented by an n -qubit register $|i\rangle_n$, to a fixed value K . We use n ancilla qubits $|a_1, \dots, a_n\rangle$, a classical array $t[n]$ holding the precomputed binary value of K 's two's complement and a qubit $|c\rangle$ which will hold the result of the comparison with $|c\rangle = 1$ if $|i\rangle \geq K$. For each qubit $|i_k\rangle$, with $k \in \{1, \dots, n\}$, we use a Toffoli gate to compute the carry at position k if $t[k] = 1$ and a logical OR, see Fig. 5, if $t[k] = 0$. For $k = 1$, we only need to use a CNOT on $|i_1\rangle$ if $t[1] = 1$. In the circuit above, only one of two unitaries in a dotted box needs to be added to the circuit, depending on the value of $t[k]$ at each qubit. The last carry qubit $|a_n\rangle$ is then used to compute the final result of the comparison in qubit $|c\rangle$.

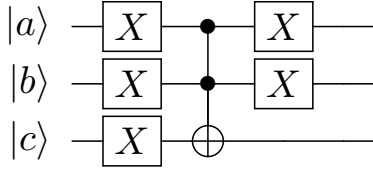


FIG. 5. Circuit that computes the logical OR between qubits $|a\rangle$ and $|b\rangle$ into qubit $|c\rangle$.

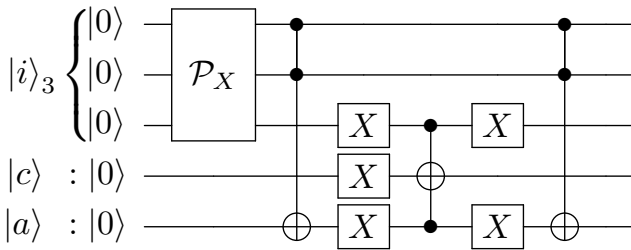


FIG. 6. Quantum circuit that sets a comparator qubit $|c\rangle$ to $|1\rangle$ if the value represented by $|i\rangle_3$ is larger than a strike $K = 1.9$, for the spot distribution in Fig. 3. The unitary \mathcal{P}_X represents the set of gates that load the probability distribution in Eq. (8). An ancilla qubit $|a\rangle$ is needed to perform the comparison. It is uncomputed at the end of the circuit.

This shows us that we needed $g_0 = \pi/4 - c$ to used the identity $\sum_i p_i = 1$ to recover $\mathbb{E}[\max(0, i - K)]$ up to a scaling factor and a constant. From this last equality we

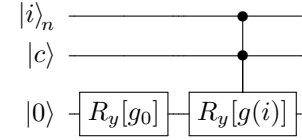


FIG. 7. Circuit that creates the state in Eq. (9). We apply this circuit directly after the comparator circuit shown in Fig. 6. The multi-controlled y -rotation is the gate shown in Fig. 2 controlled by the ancilla qubit $|c\rangle$ that contains the result of the comparison between i and K .

recover the expected payoff of the option given the probability distribution of the underlying asset. We should note that the fair value of the option requires appropriately discounting the expected payoff of the option to today, but as the discounting can be performed after the expectation value has been calculated we omit it from our discussion for simplicity. We check the correctness of our approach by running amplitude estimation using Qiskit [34] on the overall circuit produced by the elements described in this section, and verifying the convergence to the analytically computed value or classical Monte Carlo estimate. An illustration of the convergence of a European call option with increasing evaluation qubits is shown in Fig. 8.

A straightforward extension of the analysis above yields a pricing model for a European put option, whose payoff $f(i) = \max(0, K - i)$ is equivalent to Eq. (11) with $a_> = b_> = 0$, $a_< = -1$, and $b_< = K$.

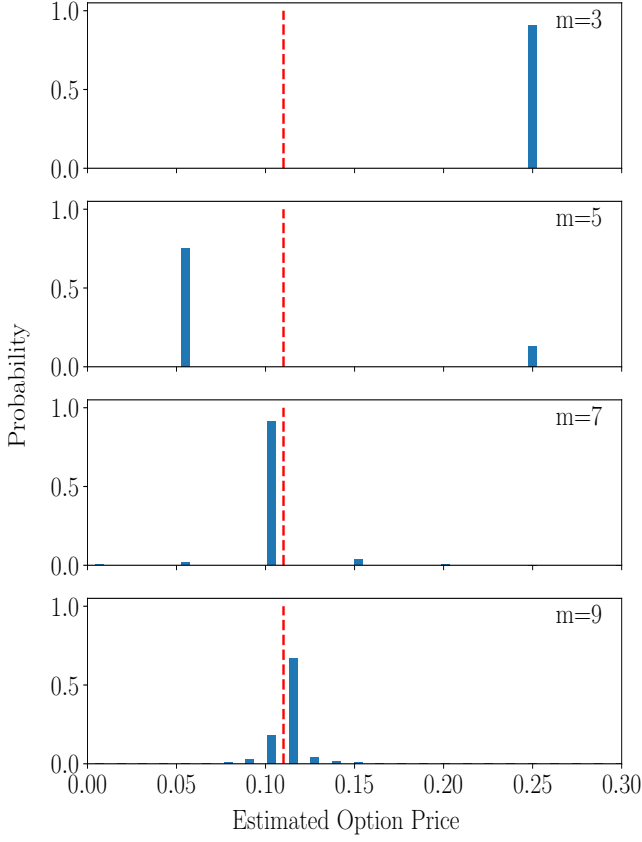


FIG. 8. Results from applying amplitude estimation (Sec. III B) on the quantum circuit of the European call option of Fig. 3, on a simulated quantum device with $m \in \{3, 5, 7, 9\}$. The red dashed line corresponds to the (undiscounted) analytical value for this option, calculated using the Black-Scholes-Merton model. We limit the range of possible option values shown to $[0, 0.3]$ to illustrate the convergence of the estimation, as the cumulative probability in the windows shown exceeds 90% in each case.

2. Portfolios of options

Various popular trading and hedging strategies rely on entering multiple option contracts at the same time instead of individual call or put options and as such, these strategies allow an investor to effectively construct a payoff that is more complex than that of vanilla options. For example, an investor that wants to profit from a volatile asset without picking a direction of where the volatility may drive the asset's price, may choose to enter a *straddle* option strategy, by buying both a call and a put option on the asset with the same expiration date and strike. If the underlying asset moves sharply up to expiration date, the investor can make a profit regardless of whether it moves higher or lower in value. Alternatively, the investor may opt for a *butterfly* option strategy by entering four appropriately structured option contracts with different strikes simultaneously. Because these option strategies give rise to piecewise linear payoff func-

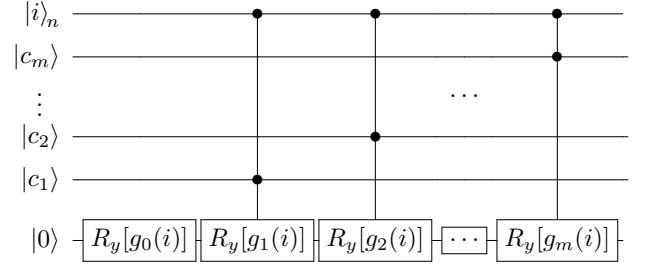


FIG. 9. Quantum circuit that implements the multi-controlled Y-rotations for a portfolio of options with m strike prices.

tions, the methodology described in the previous section can be extended to calculate the fair values of these option portfolios.

In order to capture the structure of such option strategies, we can think of the individual options as defining one or more effective strike prices K_j and add a linear function $f_j(S) = a_j S + b_j$ between each of these strikes. For example, to price an option strategy with the payoff function

$$f_s(S) = \max(0, S - K_1) - \max(0, S - K_2), \quad (15)$$

which corresponds to a call spread (the option holder has purchased a call with strike K_1 and sold a call with strike K_2), we use the functions f_0 , f_1 , and f_2 such that

$$f_s(S) = \begin{cases} f_0(S) & S < K_1, \\ f_0(S) + f_1(S) & K_1 \leq S < K_2, \\ f_0(S) + f_1(S) + f_2(S) & K_2 \leq S. \end{cases} \quad (16)$$

To match Eq. (15) with Eq. (16) we set $f_0(S) = 0$, $f_1(S) = S - K_1$ and $f_2(S) = -S + K_2$. In general, to price a portfolio of options with m effective-strike prices K_1, \dots, K_m and $m+1$ functions $f_0(S), \dots, f_m(S)$ we need an ancilla qubit per strike to indicate if the underlying has reached the strike. This allows us to generalize the discussion from Sec. IV A 1. We apply a multi-controlled Y-rotation with angle $g_j(i)$ if $i \geq K_j$ for each strike K_j with $j \in \{1, \dots, m\}$. The rotation $g_0(i)$ is always applied, see the circuit in Fig. 9. The functions $g_j(i)$ are determined using the same procedure as in Sec. IV A 1.

B. Multi-asset and path-dependent options

In this section we show how to price options with path-dependent payoffs as well as options on more than one underlying asset. In these cases, the payoff function depends on a multivariate distribution of random variables $\{S_j\}$ with $j \in \{1, \dots, d\}$. The S_j 's may represent one or several assets at discrete moments in time or a basket of assets at the option maturity. In both cases, the probability distribution of the random variables S_j are truncated to the interval $[S_{j,\min}, S_{j,\max}]$ and discretized using

2^{n_j} points so that they can be represented by d quantum registers where register j has n_j qubits. Thus, the multivariate distribution is represented by the probabilities p_{i_1, \dots, i_d} that the underlying has taken the values i_1, \dots, i_d with $i_j \in \{0, \dots, 2^{n_j} - 1\}$. The quantum state that represents this probability distribution, a generalization of Eq. (8), is

$$|\psi\rangle_n = \sum_{i_1, \dots, i_d} \sqrt{p_{i_1, \dots, i_d}} |i_1\rangle_{n_1} \otimes \dots \otimes |i_d\rangle_{n_d}, \quad (17)$$

with $n = \sum_j n_j$. Various types of options, such as Asian options or basket options, require us to compute the sum of the random variables S_j . The addition of the values in two quantum registers $|a, b\rangle \rightarrow |a, a+b\rangle$ may be calculated in quantum computers with adder circuits based on CNOT and Toffoli gates [35–37]. To this end we add an extra qubit register with n' qubits to serve as an accumulator. By recursively applying adder circuits we perform the transformation $|\psi\rangle_n |0\rangle_{n'} \rightarrow |\phi\rangle_{n+n'}$ where $|\phi\rangle_{n+n'}$ is given by

$$\sum_{i_1, \dots, i_d} \sqrt{p_{i_1, \dots, i_d}} |i_1\rangle_{n_1} \otimes \dots \otimes |i_d\rangle_{n_d} \otimes |i_1 + \dots + i_d\rangle_{n'}. \quad (18)$$

Here circuit optimization may allow us to perform this computation in-place to minimize the number of qubit registers needed. Now, we use the methods discussed in the previous section to encode the option payoffs into the quantum circuit.

1. Basket Options

A European style basket option is an extension of the single asset European option discussed in Sec. IV A, only now the payoff depends on a weighted sum of d underlying assets. A call option on a basket has the payoff profile

$$f(S_{\text{basket}}) = \max(0, S_{\text{basket}} - K) \quad (19)$$

where $S_{\text{basket}} = \vec{w} \cdot \vec{S}$, for basket weights $\vec{w} = [w_1, w_2, \dots, w_d]$, $w_i \in [0, 1]$, underlying asset prices at option maturity $\vec{S} = [S_1, S_2, \dots, S_d]$ and strike K . In the BSM model, the underlying asset prices are described by a multivariate lognormal distribution with probability density function [38]

$$P(\vec{S}) = \frac{\exp(-\frac{1}{2}(\ln S - \mu)^T \Sigma^{-1}(\ln S - \mu))}{(2\pi)^{d/2} (\det \Sigma)^{1/2} \prod_{i=1}^d S_i}, \quad (20)$$

where $\ln S = (\ln S_1, \ln S_2, \dots, \ln S_d)^T$ and $\mu = (\mu_1, \mu_2, \dots, \mu_d)^T$, where each μ_i is the lognormal distribution parameter for each asset defined in the caption of

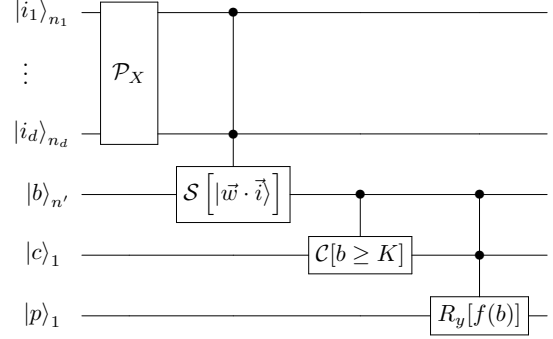


FIG. 10. Schematic of the circuit that encodes the payoff of a basket call option of d underlying assets into the amplitude of a payoff qubit $|p\rangle$. First, a unitary \mathcal{P}_X loads the multivariate distribution of Eq. (20) into d registers $|i_1\rangle_{n_1} \dots |i_d\rangle_{n_d}$ using the methods described in Sec. III A. The weighted sum operator \mathcal{S} , see Appendix A, calculates the weighted sum $|w_1 \cdot i_1 + \dots + w_d \cdot i_d\rangle$ into a register $|b\rangle_{n'}$ with n' qubits, where n' is large enough to hold the maximum possible sum. The comparator circuit \mathcal{C} sets a comparator qubit $|c\rangle$ to $|1\rangle$ if $b \geq K$. Lastly, controlled-Y rotations are used to encode the option payoff $f(b) = \max(0, b - K)$ into the payoff qubit using the method shown in Fig. 7, controlled by the comparator qubit $|c\rangle$.

Fig. 3. Σ is the $d \times d$ positive-definite covariance matrix of the d underlyings

$$\Sigma = T \begin{bmatrix} \sigma_1^2 & \rho_{12}\sigma_1\sigma_2 & \dots & \rho_{1d}\sigma_1\sigma_d \\ \rho_{21}\sigma_2\sigma_1 & \sigma_2^2 & \dots & \rho_{2d}\sigma_2\sigma_d \\ \vdots & \vdots & \ddots & \vdots \\ \rho_{d1}\sigma_d\sigma_1 & \dots & \dots & \sigma_d^2 \end{bmatrix} \quad (21)$$

with σ_i the volatility of the i th asset, $-1 \leq \rho_{ij} \leq 1$ the correlation between assets i and j and T the time to maturity.

The quantum circuit for pricing a European style basket call option is then analogous to the single asset case, with an additional unitary to compute the weighted sum of the uncertainty registers $|i_1\rangle_{n_1} \dots |i_d\rangle_{n_d}$ before encoding the comparator and payoff circuits, with the accumulator register $|b\rangle_{n'} = |i_1 + \dots + i_d\rangle_{n'}$ acting as the control. An implementation of the circuit that performs the weighted sum operator is presented in more detail in Appendix A. A schematic of these components is shown in Fig. 10.

2. Asian Options

We now examine arithmetic average Asian options which are single-asset, path-dependent options whose payoff depends on the price of the underlying asset at multiple time points before the option's expiration date. Specifically, the payoff of an Asian call option is given by

$$f(\bar{S}) = \max(0, \bar{S} - K) \quad (22)$$

where K is the strike price, \bar{S} is the arithmetic average of the asset's value over a pre-defined number of points d between 0 and the option maturity T

$$\bar{S} = \frac{1}{d} \sum_{t=1}^d S_t. \quad (23)$$

The probability distribution of asset prices at time t will again be lognormal with probability density function

$$P(S_t) = \frac{1}{S_t \sigma \sqrt{2\pi \Delta t}} e^{-\frac{(\ln S_t - \mu_t)^2}{2\sigma^2 \Delta t}} \quad (24)$$

with $\mu_t = (r - 0.5\sigma^2)\Delta t + \ln(S_{t-1})$ and $\Delta t = T/d$. We can then use the multivariate distribution in Eq. (20), with \vec{S} now a d -dimensional vector of asset prices at time points $[t_1 \dots t_d]$, instead of distinct underlying prices at maturity T . As we are not considering multiple underlying assets that could be correlated, the covariance matrix is diagonal $\Sigma = \Delta t[\text{diag}(\sigma^2, \dots, \sigma^2)]$. An illustration of the probability density function used for an asset defined on two time steps is shown in Fig. 11.

We now prepare the state $|\psi\rangle_n$, see Eq. (17), where each register represents the asset price at each time step up to maturity. Using the weighted sum operator of Appendix A with equal weights $1/d$, we then calculate the average value of the asset until maturity T , see Eq. (23), into a register $|\bar{S}\rangle$

$$\underbrace{|i_1\rangle}_{\Delta t} \underbrace{|i_2\rangle}_{2\Delta t} \dots \underbrace{|i_d\rangle}_T \mapsto |\bar{S}\rangle = \left| \frac{1}{d} \sum_{t=1}^d S_t \right\rangle. \quad (25)$$

Finally, we use the same comparator and rotation circuits that we employed for the basket option illustrated in Fig. 10 to load the payoff of an arithmetic average Asian option into the payoff qubit $|p\rangle$.

3. Barrier Options

Barrier options are another class of popular option types whose payoff is similar to vanilla European Options, but they become activated or extinguished if the underlying asset crosses a pre-determined level called the *barrier*. In their simplest form, there are two general categories of barrier options

- **Knock-Out:** The option expires worthless if the underlying asset crosses a certain price level before the option's maturity.

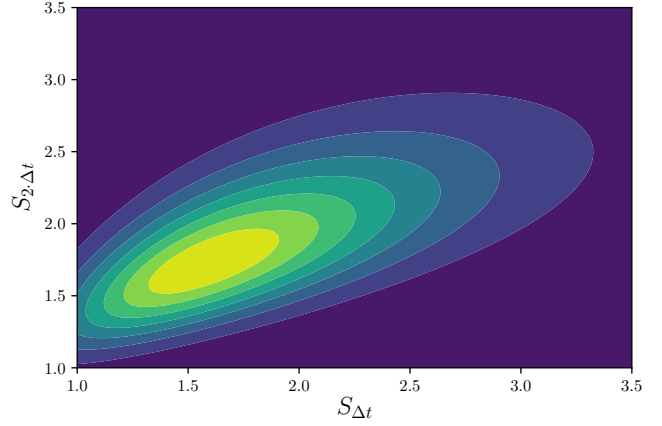


FIG. 11. Probability density function of a multivariate lognormal distribution, see Eq. (20), for the asset shown in Fig. 3 defined on two time steps $t = \Delta t = T/2$ and $t = 2\Delta t = T$

- **Knock-In:** The option has no value unless the underlying asset crosses a certain price level before maturity.

If the required barrier event for the option to have value at maturity occurs, the payoff then depends only on the value of the underlying asset at maturity and not on the path of the asset until then. If we consider a Knock-In barrier option and label the barrier level B , we can write the option's payoff as

$$f(S) = \begin{cases} \max(0, S_T - K) & \text{if } \exists t \text{ s.t. } S_t \geq B \\ 0 & \text{otherwise} \end{cases} \quad (26)$$

where T is the time to maturity, S_t the asset price at time t with $0 < t \leq T$ and K the option strike.

To construct a quantum circuit to price a Knock-In barrier option, we use the same method as for the Asian option where T is divided into d equidistant time intervals with $\Delta t = T/d$, and use registers $|i_1\rangle_{n_1} |i_2\rangle_{n_2} \dots |i_d\rangle_{n_d}$ to represent the discretized range of asset prices at time $t \in \{\Delta t, 2\Delta t, \dots, d \cdot \Delta t = T\}$. The probability distribution of Eq. (24) is used again to create the state $|\psi\rangle_n$ in Eq. (17).

To capture the path dependence introduced by the barrier, we use an additional d -qubit register $|b\rangle_d$ to monitor if the barrier is crossed. Each qubit $|b_t\rangle$ in $|b\rangle_d$ is set to $|1\rangle$ if $|i_t\rangle_{n_t} \geq B$. An ancilla qubit $|b_1\rangle$ is set to $|1\rangle$ if the barrier has been crossed in at least one time step. This is done by computing the logical OR, see Fig. 5, of every qubit in $|b\rangle_d$ and storing the result in the ancilla

$$|b_1 b_2 \dots b_d\rangle |0\rangle \mapsto |b_1 b_2 \dots b_d\rangle |b_1 \vee b_2 \dots \vee b_d\rangle. \quad (27)$$

This is computed with X (NOT) and Toffoli gates and $d - 2$ ancilla qubits. The ancilla qubit $|b_1\rangle$ is then used as a control for the payoff rotation into the payoff qubit, effectively knocking the option *in*. For Knock-Out barrier

options, we can follow the same steps and apply an X gate to the ancilla barrier qubit before using it as control, in this manner knocking the option *out* if the barrier level has been crossed. A circuit displaying all the components required to price a Knock-In barrier option is shown in Fig. 12. Results from amplitude estimation on a barrier option circuit using a quantum simulator are shown in Fig. 13.

Even though we have focused our attention on barrier options where the barrier event is the underlying asset crossing a barrier from below, we can use the same method to price barrier options where barrier events are defined as the asset crossing the value from above. This only requires changing the comparator circuits to compute $S_t \leq B$ in the barrier register $|b\rangle_d$.

V. QUANTUM HARDWARE RESULTS

In this section we show results for a European call option evaluated on quantum hardware. We use three qubits in total, two to represent the uncertainty and one to encode the payoff.

We consider a log-normal random distribution with $S_0 = 2$, $\sigma = 13\%$, $r = 5\%$, and $T = 40/365$, (see Fig. 3) and truncate the distribution to the interval defined by three standard deviations around the mean. With two qubits encoding this distribution, the possible values are $[1.21, 1.74, 2.28, 2.81]$, represented by $|00\rangle, \dots, |11\rangle$, with corresponding probabilities 0.1%, 55.4%, 42.5%, and 1.9%. We set the strike price to $K = 1.74$. The resulting exact expected payoff of this option is equal to 0.256.

We now show how to construct the operator \mathcal{A} that is required for AE. The log-normal distribution can be loaded using only two CNOT gates and three single qubit Y-rotations. The corresponding circuit is illustrated in Fig. 14.

To encode the payoff function we exploit the small number of qubits and apply a uniformly controlled Y-rotation instead of the generic construction using comparators introduced in Sec. IV. A uniformly controlled Y-rotation implements a different rotation angle for each possible state of the control qubits, i.e., it applies the operation

$$|i\rangle_n |0\rangle \mapsto |i\rangle_n R_y(\theta_i) |0\rangle, \quad (28)$$

for given angles θ_i , $i = 0, \dots, 2^n - 1$. For $n = 2$, this operation can be efficiently implemented using four CNOT gates and four single qubit Y-rotations [25, 39]. Although we use a uniformly controlled rotation, the rotation angles are constructed in the same way described in Sec. IIID. We use an approximation scaling of $c = 0.25$ and the resulting angles are $[\theta_0, \dots, \theta_3] = [1.1781, 1.1781, 1.5708, 1.9635]$, which shows the piecewise linear structure of the payoff function.

The total resulting circuit requires six CNOT gates and seven single-qubit Y-rotations, see Fig. 14. Since we use

uniformly controlled gates, we do not need any ancilla qubits.

We now execute $\mathcal{A}|0\rangle_3$ on the *IBM Q Tokyo* 20-qubit device and estimate the probability P_1 of measuring $|1\rangle$ in the objective qubit using 8192 shots. The connectivity of *IBM Q Tokyo* allows to choose three fully connected qubits, thus, no additional swaps are required [34]. The experiment on quantum hardware results in $P_1^Q = 40.9\%$ compared to the analytic result of $P_1^{SIM} = 39.7\%$ obtained using a simulator. This is a relative error of only 3%, i.e., the real hardware accurately reproduces the desired quantum state. Mapped to the expected payoff of the option, this corresponds to 0.287, i.e., a relative error of 12%. The increased error results from applying the inverse approximation scaling $1/c = 4$.

Evaluating P_1 for the state $\mathcal{A}|0\rangle_3$ on a quantum computer corresponds to a classical Monte Carlo simulation, where the actual samples are generated by a quantum computer. This does not provide a conceptual speed-up but demonstrates that generating quantum samples is possible on current quantum hardware.

It has been shown that a quadratic speed-up can also be realized by measuring $Q^k \mathcal{A}|0\rangle$ for different k combined with a maximum likelihood estimation [40], leading to significantly shorter circuits than the original implementation of AE. Thus, to demonstrate the improved scaling, we also evaluate $Q\mathcal{A}|0\rangle_3$ on real hardware. After optimization, the resulting circuit consists of 24 CNOT gates and 28 single-qubit gates. However, this is slightly out of reach for current hardware as our measurement $P_1^Q = 60\%$ deviates significantly from $P_1^{SIM} = 79\%$.

VI. CONCLUSION

We have presented a methodology and the quantum circuits to price options and option portfolios on a gate-based quantum computer. We showed how to account for some of the more complex features present in exotic options such path-dependency with barriers and averages. The results that we show are available in Qiskit Finance [34]. Future work may involve calculating the price derivatives [41] with a quantum computer. Pricing options relies on AE. This quantum algorithm allows a quadratic speed-up compared to traditional Monte Carlo simulations. However, larger quantum hardware capable of running deeper quantum circuits with more qubits than the currently available quantum computers is needed to price the typical portfolios seen in the financial industry. Additionally, more research is needed to reduce the quantum resources required by the algorithms. Future work could focus on reducing the number of quantum registers in our implementation by performing some of the computation in-place.

Algorithms based on AE will most likely require a universal fault-tolerant quantum computer [42]. However, research to improve the algorithms is ongoing [43–45] and methods such as error mitigation are being devel-

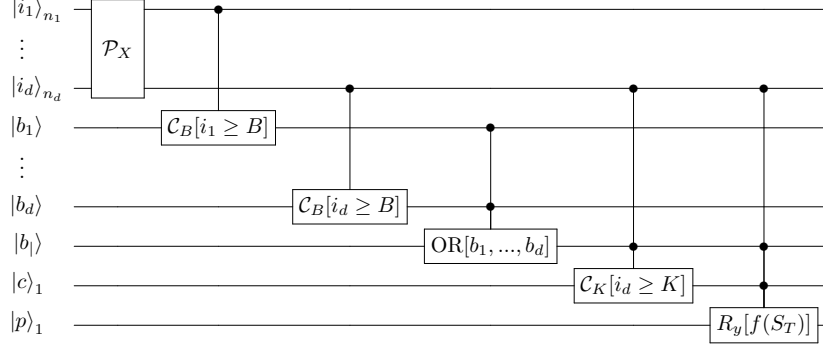


FIG. 12. Circuit that encodes the payoff of a Knock-In barrier option in the state of an ancilla qubit $|p\rangle_1$. The unitary operator \mathcal{P}_X is used to initialize the state of Eq. (17). Comparator circuits \mathcal{C}_B are used to set a barrier qubit b_j for all $j \in [1, d]$ if the asset price represented by $|i_j\rangle$ crosses the barrier B . The logical OR of all b_j qubits is computed into ancilla $|b_1\rangle$. The strike comparator circuit \mathcal{C}_K sets the comparator qubit $|c_1\rangle$ to $|1\rangle$ if the asset price at maturity $|i_d\rangle \geq K$. Finally, Y-rotations encode the payoff qubit $|p\rangle_1$, controlled on $|i_d\rangle$, the strike qubit $|c_1\rangle$ and the barrier qubit $|b_1\rangle$ which is $|1\rangle$ only if the asset price has crossed the barrier at least once before maturity.

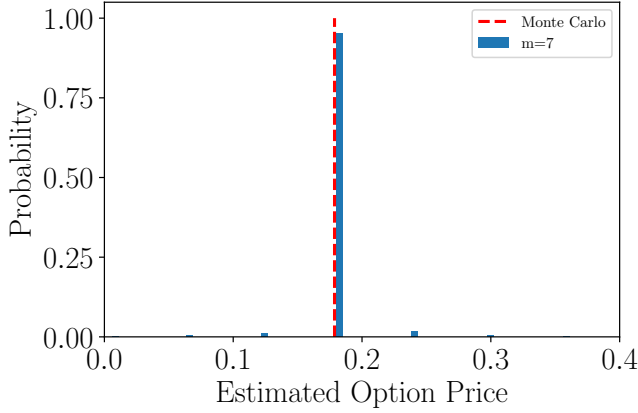


FIG. 13. Estimated option price for a barrier option using amplitude estimation on a quantum simulator. The option is defined on the asset of Fig. 3 with two timesteps $T/2$ and $T = 300/365$ and 2 qubits used to represent the uncertainty per timestep. The option strike is $K = 1.9$ and a barrier was added at $B = 2.0$ on both timesteps. The red dotted line is the (undiscounted) value of the option calculated with classical Monte Carlo and 100,000 paths and the blue bars show the estimated option values using amplitude estimation with $m = 7$ sampling qubits.

VII. ACKNOWLEDGMENTS

Opinions and estimates constitute our judgment as of the date of this Material, are for informational purposes only and are subject to change without notice. This Material is not the product of J.P. Morgan's Research Department and therefore, has not been prepared in accordance with legal requirements to promote the independence of research, including but not limited to, the prohibition on the dealing ahead of the dissemination of investment research. This Material is not intended as research, a recommendation, advice, offer or solicitation for the purchase or sale of any financial product or service, or to be used in any way for evaluating the merits of participating in any transaction. It is not a research report and is not intended as such. Past performance is not indicative of future results. Please consult your own advisors regarding legal, tax, accounting or any other aspects including suitability implications for your particular circumstances. J.P. Morgan disclaims any responsibility or liability whatsoever for the quality, accuracy or completeness of the information herein, and for any reliance on, or use of this material in any way. Important disclosures at: www.jpmorgan.com/disclosures

oped to extend the reach of noisy quantum computers [8]. For instance, Ref. [40] shows that it is possible to retain the speed-up from AE with shorter quantum circuits by replacing the quantum phase estimation in AE by a classical maximum likelihood estimation on multiple runs with an increasing number of quantum samples.

IBM, IBM Q, Qiskit are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product or service names may be trademarks or service marks of IBM or other companies.

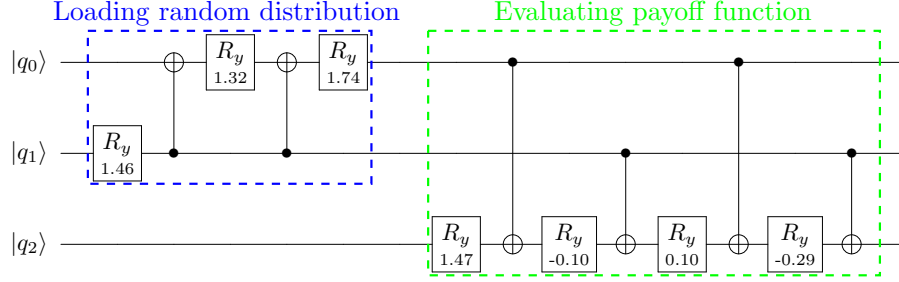


FIG. 14. The \mathcal{A} operator of the considered European call option: first, the 2-qubit approximation of a log-normal distribution is loaded, and second, the piecewise linear payoff function is applied to last qubit. This operator can be used within amplitude estimation to evaluate the expected payoff of the corresponding option.

Appendix A: Circuit implementation of weighted sum operator

1. Weighted sum of single qubits

In this appendix, we demonstrate an implementation of the weighted sum operator on a quantum circuit. The weighted sum operator \mathcal{S} computes the arithmetic sum of the values of n qubits $|a\rangle_n = |a_1 \dots a_n\rangle$ weighted by n classically defined non-negative integer weights $\omega = (\omega_1, \omega_2, \dots, \omega_n)$, and stores the result into another m -qubit register $|s\rangle_m = |s_1 \dots s_m\rangle$ initialized to $|0\rangle_m$. In other words,

$$\mathcal{S} |a\rangle_n |0\rangle_m = |a\rangle_n \left| \sum_{i=1}^n \omega_i a_i \right\rangle_m, \quad (\text{A1})$$

where

$$m = \left\lceil \log_2 \left(\sum_{i=0}^n \omega_i \right) \right\rceil + 1. \quad (\text{A2})$$

The choice of m ensures that the sum register $|s\rangle_m$ is large enough to hold the largest possible weighted sum, i.e. the sum of all weights. Alternatively, we can write the weights in the form of a binary matrix $\Omega = (\Omega_{i,j}) \in \{0,1\}^{n \times n^*}$, where the i -th row in Ω is the binary representation of weight ω_i and $n^* = \max_{i=1}^n n_i$. We use the convention that less significant digits have smaller indices, so $|s_1\rangle$ and $\Omega_{i,1}$ are the least significant digits of the respective binary numbers. Using this binary matrix representation, \mathcal{S} is to add the i -th qubit $|a_i\rangle$ of the state register to the j -th qubit $|s_j\rangle$ of the sum register if and only if $\Omega_{i,j} = 1$. Depending on the values of the weights, an additional quantum register may be necessary to temporarily store the carries during addition operations. We use $|c_j\rangle$ to denote the ancilla qubit used to store the carry from adding a digit to $|s_j\rangle$. These ancilla qubits are initialized to $|0\rangle$ and will be reset to their initial states at the end of the computation.

Based on the above setup, we build quantum circuits for the weighted sum operator using three elementary

gates: X (NOT), controlled-NOT, and the Toffoli gate (controlled-controlled-NOT). These three gates suffice to build any Boolean function [35]. Starting from the first column in Ω , for each column j , we find all elements with $\Omega_{i,j} = 1$ and add the corresponding state qubit $|a_i\rangle$ to $|s_j\rangle$. The addition of two qubits involves three operations detailed in Fig. 15: (a) computation of the carry using a Toffoli gate (\mathcal{M}), (b) computation of the current digit using a controlled-NOT (\mathcal{D}), (c) reset of the carry computation using two X gates and one Toffoli gate ($\bar{\mathcal{M}}$). When adding $|a_i\rangle$ to the j -th qubit of the sum register, the computation starts by applying \mathcal{M} and then \mathcal{D} to $|a_i\rangle$, $|s_j\rangle$ and $|c_j\rangle$, which adds $|a_i\rangle$ to $|s_j\rangle$ and stores the carry into $|c_j\rangle$. Then, using the same two operations, it adds the carry $|c_j\rangle$ to the next sum qubit $|s_{j+1}\rangle$ with carry recorded in $|c_{j+1}\rangle$. The process is iterated until all carries are handled. Finally, it resets the carry qubits by applying $\bar{\mathcal{M}}$ in reverse order of the carry computation. We reset the carry qubits in order to reuse them in later computations if necessary.

In general, we need $\max(k-2, 0)$ carry qubits to compute the addition of $|a_i\rangle$ on $|s_j\rangle$, where $k \geq 1$ is the smallest integer satisfying

$${}_k \langle 1 | \rho_{j,j+k-1}^s | 1 \rangle_k = 0, \quad (\text{A3})$$

where $\rho_{j,j+k-1}^s$ is the density matrix corresponding to $|s_j \dots s_{j+k-1}\rangle$. In the $k=1$ case, i.e. $|s_j\rangle = 0$, the computation is reduced to “copying” $|a_i\rangle$ to $|s_j\rangle$ using the bit addition operator \mathcal{D} , and no carries would be produced. For $k \geq 2$, Eq. (A3) guarantees no carries from $|s_{j+k-1}\rangle$ and beyond. Therefore we can directly compute the carry from $|s_{j+k-2}\rangle$ into $|s_{j+k-1}\rangle$ without worrying about additional carries. This eliminates the need for an ancilla qubit $|c_{j+k-2}\rangle$, and hence the number of carry qubits needed is $k-2$. To further reduce the number of ancilla qubits, we can use any sum qubit $|s_j\rangle = |0\rangle$ during the computation. In our case, since we are processing Ω column by column, all sum qubits more significant than $|s_{j+k-1}\rangle$ would be $|0\rangle$. In other words, we have the last $m - (j+k-1)$ sum qubits usable as carry qubits in the computation described above.

As the weights are known at the time of building the

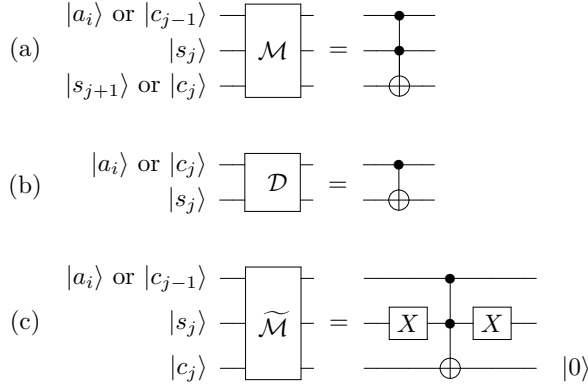


FIG. 15. Three component gates used to construct the weighted sum operator \mathcal{S} . (a) The carry operator \mathcal{M} consisting of one Toffoli gate, which computes the carry from adding $|a_i\rangle$ (or $|c_{j-1}\rangle$) and $|s_j\rangle$ into $|s_{j+1}\rangle$ or $|c_j\rangle$. (b) The bit addition operator \mathcal{D} consisting of one controlled-NOT gate, which adds the state qubit $|a_i\rangle$ or the carry qubit from the previous digit $|c_{j-1}\rangle$ to the sum qubit $|s_j\rangle$. (c) The carry reset operator $\tilde{\mathcal{M}}$ consisting of two X gates and one Toffoli gate, which resets the carry qubit $|c_j\rangle$ back to $|0\rangle$.

circuit, the possible states that $|s\rangle_m$ can have before each addition of a state qubit $|a_i\rangle$ are also computable. Since we are adding $|a_i\rangle$ to $|s\rangle_m$ starting from the least significant bit, k equals the bit length of the maximum possible sum on $|s_j \dots s_m\rangle$ after adding $|a_i\rangle$ to $|s_j\rangle$. In other words,

$$k = \log_2 \left\lceil \sum_{\substack{u \leq i, \text{ or} \\ v \leq j}} \frac{\Omega_{u,v}}{2^{j-v}} \right\rceil + 1. \quad (\text{A4})$$

Therefore, the number of carry operations and additional ancilla qubits required for each addition of $|a_i\rangle$ can be determined. The term in the $\lceil \cdot \rceil$ in Eq. (A4) is upper-bounded by

$$\sum_{\substack{u \leq i, \text{ or} \\ v \leq j}} \frac{\Omega_{u,v}}{2^v} \leq \sum_{j=1}^m \frac{n_{\max}}{2^{j-1}} < 2n_{\max} \leq 2n, \quad (\text{A5})$$

where $n_{\max} = \max_{j=1}^m \sum_{i=1}^n n\Omega_{i,j}$ is the maximum number of 1's in a column of Ω . It immediately follows that the number of non-trivial carry operations (i.e. carry operations that requires $\tilde{\mathcal{M}}$) required to add $|a_i\rangle$ to $|s_j \dots s_m\rangle$ is upper-bounded by

$$k - 2 < \log_2 \lfloor n_{\max} \rfloor \leq \log_2 \lfloor n \rfloor, \quad (\text{A6})$$

and the number of ancilla qubits required for the entire implementation of \mathcal{S} is at most the upper bound for $k-2$, since we may use some sum qubits as carries. In other words, the number of ancilla qubits required for \mathcal{S} grows at most logarithmically with the number of state qubits n .

2. Sum of multi-qubit integers

The weighted sum operator \mathcal{S} can be used to calculate the sum of d multi-qubit positive integers on a quantum register. To do that we first prepare the input register in the state

$$|a\rangle_n = |a_1^{(1)} \dots a_{n_1}^{(1)} \dots a_1^{(d)} \dots a_{n_d}^{(d)}\rangle, \quad n = \sum_{i=1}^d n_i, \quad (\text{A7})$$

where $|a_1^{(i)} \dots a_{n_i}^{(i)}\rangle, i \in [1, d]$ is the binary representation of the i -th integer to sum with n_i qubits, least significant figure first. Then we set the weights as

$$\omega = (2^0, \dots, 2^{n_1-1}, \dots, 2^0, \dots, 2^{n_d-1}), \quad (\text{A8})$$

or equivalently,

$$\Omega_{n \times n^*} = (I_{n_1 \times n^*}^T, \dots, I_{n_d \times n^*}^T)^T, \quad (\text{A9})$$

where $I_{n_i \times n^*} = (I_{n_i}, 0_{n_i \times (n^* - n_i)})$, $i \in [1, d]$ and I_{n_i} is the n_i -dimensional identity matrix. Now if we build a weighted sum operator based on the weights in Eq. (A8) and apply it on the input state qubits in Eq. (A7), we would have the sum of the d integers in $|s\rangle_m$.

Fig. 16 shows an example circuit computing the sum of two 3-digit binary numbers represented on a 6-qubit quantum register $|a\rangle_3 |b\rangle_3$, and storing the result into a 4-qubit register $|s\rangle_4$. The circuit is implemented by a weighted sum operator \mathcal{S} with weights $\omega = (1, 2, 4, 1, 2, 4)$. The operator consists of 6 controlled-NOT (\mathcal{D}) gates and 3 Toffoli (\mathcal{M}) gates. The 9 gates are grouped in three groups of 3, as is shown in Fig. 16 by dashed boxes. Each group computes the sum of the bits $|a_j\rangle$ and $|b_j\rangle$ into $|s_j\rangle$ and the carry into $|s_{j+1}\rangle$. Note that separate carry qubits are not required, therefore no carry reset operators $\tilde{\mathcal{M}}$ are used. In fact, using the above construction for \mathcal{S} , no extra carry qubits will be required for the addition of any two binary numbers. In general, \mathcal{S} requires at most $\lfloor \log_2 d \rfloor$ ancilla qubits for carrying operations, which directly comes from Eq. (A6).

3. Weighted sum of multi-qubit integers

In addition to summing up d integers equally, a weight w_i may also be added to each integer $a^{(i)}$. In that case, the weight matrix would be

$$\Omega = (w_1 \cdot I_{n_1 \times n^*}^T, \dots, w_d \cdot I_{n_d \times n^*}^T)^T. \quad (\text{A10})$$

In the case where w_i are not integers, we can rescale the values represented on the quantum register by a common factor to make all weights integers. For example, if we are adding two numbers with weights 0.2 and 0.8, we could use integer weights of $w_1 = 1$ and $w_2 = 4$ instead, and reinterpret the resulting sum in postprocessing by dividing it by 5.

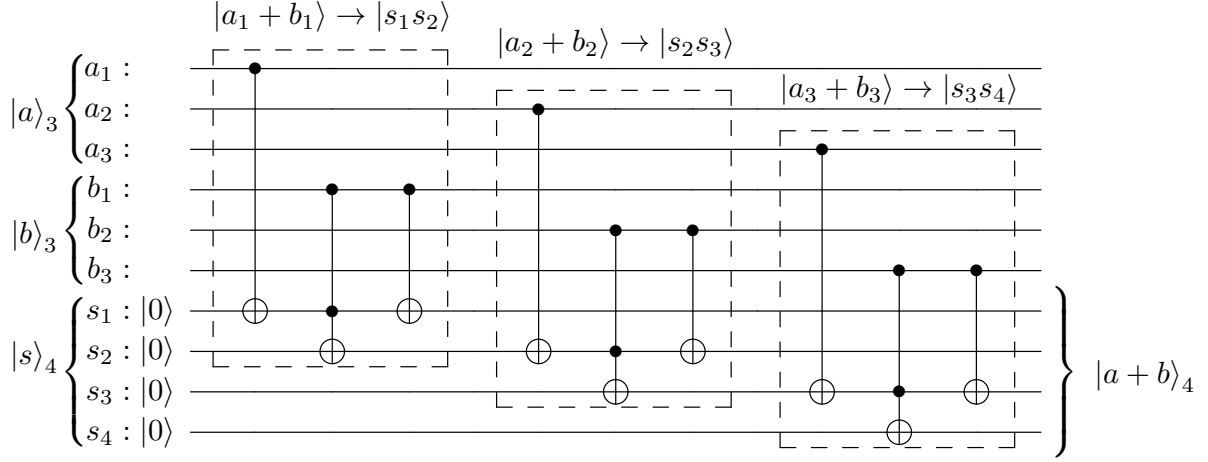


FIG. 16. A circuit computing the sum of binary numbers $|a\rangle_3$ and $|b\rangle_3$ into $|s\rangle_4$ implemented using the weighted sum operator with weights $\omega = (1, 2, 4, 1, 2, 4)$.

-
- [1] John C. Hull, *Options, futures, and other derivatives*, 6th ed. (Pearson Prentice Hall, Upper Saddle River, NJ [u.a.], 2006).
- [2] Fischer Black and Myron Scholes, "The pricing of options and corporate liabilities," *Journal of Political Economy* **81**, 637–654 (1973).
- [3] Bruno Dupire, "Pricing with a smile," *Risk Magazine*, 18–20 (1994).
- [4] Phelim P. Boyle, "Options: A Monte Carlo approach," *Journal of Financial Economics* **4**, 323–338 (1977).
- [5] Paul Glasserman, *Monte Carlo Methods in Financial Engineering* (Springer-Verlag New York, 2003) p. 596.
- [6] Michael A. Nielsen and Isaac L. Chuang, *Cambridge University Press* (2010) p. 702.
- [7] Abhinav Kandala, Antonio Mezzacapo, Kristan Temme, Maika Takita, Markus Brink, Jerry M. Chow, and Jay M. Gambetta, "Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets," *Nature* **549**, 242 (2017).
- [8] Abhinav Kandala, Kristan Temme, Antonio D. Corcoles, Antonio Mezzacapo, Jerry M. Chow, and Jay M. Gambetta, "Error mitigation extends the computational reach of a noisy quantum processor," *Nature* **567**, 491–495 (2019).
- [9] N. Moll, P. Barkoutsos, L. S. Bishop, J. M. Chow, A. Cross, D. J. Egger, S. Filipp, A. Fuhrer, J. M. Gambetta, M. Ganzhorn, A. Kandala, A. Mezzacapo, P. Müller, W. Riess, G. Salis, J. Smolin, I. Tavernelli, and K. Temme, "Quantum optimization using variational algorithms on near-term quantum devices," *Quantum Science and Technology* **3**, 030503 (2018).
- [10] M. Ganzhorn, D.J. Egger, P. Barkoutsos, P. Ollitrault, G. Salis, N. Moll, M. Roth, A. Fuhrer, P. Mueller, S. Woerner, I. Tavernelli, and S. Filipp, "Gate-efficient simulation of molecular eigenstates on a quantum computer," *Phys. Rev. Applied* **11**, 044092 (2019).
- [11] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd, "Quantum algorithm for linear systems of equations," *Physical Review Letters* **103**, 1–8 (2009), [arXiv:0811.3171](#).
- [12] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost, "Quantum principal component analysis," *Nature Physics* **10** (2014).
- [13] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd, "Quantum machine learning," *Nature* **549**, 195–202 (2017).
- [14] Vojtech Havlicek, Antonio D. Corcoles, Kristan Temme, Aram W. Harrow, Abhinav Kandala, Jerry M. Chow, and Jay M. Gambetta, "Supervised learning with quantum-enhanced feature spaces," *Nature* **567**, 209 – 212 (2019).
- [15] Roman Orus, Samuel Mugel, and Enrique Lizaso, "Quantum computing for finance: Overview and prospects," *Reviews in Physics* **4**, 100028 (2019).
- [16] Patrick Rebentrost and Seth Lloyd, "Quantum computational finance: quantum algorithm for portfolio optimization," (2018), [arXiv:1811.03975](#).
- [17] Stefan Woerner and Daniel J. Egger, "Quantum risk analysis," *npj Quantum Information* **5**, 15 (2019).
- [18] Patrick Rebentrost, Brajesh Gupta, and Thomas R. Bromley, "Quantum computational finance: Monte carlo pricing of financial derivatives," *Phys. Rev. A* **98**, 022321 (2018).
- [19] Christa Zoufal, Aurélien Lucchi, and Stefan Woerner, "Quantum generative adversarial networks for learning and loading random distributions," (2019), [arXiv:1904.00043](#).
- [20] Ana Martin, Bruno Candelas, Angel Rodriguez-Rozas, Jose D. Martin-Guerrero, Xi Chen, Lucas Lamata, Roman Orus, Enrique Solano, and Mikel Sanz, "Towards pricing financial derivatives with an ibm quantum computer," (2019).

- [21] Gilles Brassard, Peter Hoyer, Michele Mosca, and Alain Tapp, “Quantum Amplitude Amplification and Estimation,” *Contemporary Mathematics* **305** (2002).
- [22] Robert C. Merton, “Theory of rational option pricing,” *The Bell Journal of Economics and Management Science* **4**, 141–183 (1973).
- [23] Lov Grover and Terry Rudolph, “Creating superpositions that correspond to efficiently integrable probability distributions,” (2002), [arXiv:0208112](#).
- [24] Emanuel Derman and Iraj Kani, “The volatility smile and its implied tree,” Goldman Sachs Quantitative Strategies Research Notes (1994).
- [25] Vivek V Shende, Stephen S Bullock, and Igor L Markov, “Synthesis of quantum-logic circuits,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* **25**, 1000–1010 (2006).
- [26] Martin Plesch and Caslav Brukner, “Quantum-state preparation with universal gate decompositions,” *Phys. Rev. A* **83** (2010).
- [27] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio, “Generative adversarial nets,” in *Advances in Neural Information Processing Systems 27*, edited by Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger (Curran Associates, Inc., 2014) pp. 2672–2680.
- [28] Daniel S Abrams and Colin P Williams, “Fast quantum algorithms for numerical integrals and stochastic processes,” (1999), [arXiv:9908083](#).
- [29] Ashley Montanaro, “Quantum speedup of monte carlo methods,” *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* **471** (2015).
- [30] A. Yu. Kitaev, “Quantum measurements and the Abelian Stabilizer Problem,” (1995), [arXiv:9511026](#).
- [31] Paul Glasserman, Philip Heidelberger, and Perwez Shahabuddin, “Efficient Monte Carlo Methods for Value-at-Risk,” in *Mastering Risk*, Vol. 2 (2000) pp. 5–18.
- [32] Adriano Barenco, Charles H. Bennett, Richard Cleve, David P. Divincenzo, Norman Margolus, Peter Shor, Tycho Sleator, John A. Smolin, and Harald Weinfurter, “Elementary gates for quantum computation,” *Phys. Rev. A* **52**, 3457–3467 (1995).
- [33] Steven A Cuccaro, Thomas G Draper, Samuel A Kutin, and David Petrie Moulton, “A new quantum ripple-carry addition circuit,” (2004), [arXiv:0410184](#).
- [34] Gadi Aleksandrowicz, Thomas Alexander, Panagiotis Barkoutsos, Luciano Bello, Yael Ben-Haim, David Bucher, Francisco Jose Cabrera-Hernández, Jorge Carballo-Franquis, Adrian Chen, Chun-Fu Chen, Jerry M. Chow, Antonio D. Córcoles-Gonzales, Abigail J. Cross, Andrew Cross, Juan Cruz-Benito, Chris Culver, Salvador De La Puente González, Enrique De La Torre, Delton Ding, Eugene Dumitrescu, Ivan Duran, Pieter Eendebak, Mark Everitt, Ismael Faro Sertage, Albert Frisch, Andreas Fuhrer, Jay Gambetta, Borja Godoy Gago, Juan Gomez-Mosquera, Donny Greenberg, Ikko Hamamura, Vojtech Havlicek, Joe Hellmers, Łukasz Herok, Hiroshi Horii, Shaohan Hu, Takashi Imamichi, Toshinari Itoko, Ali Javadi-Abhari, Naoki Kanazawa, Anton Karazeev, Kevin Krsulich, Peng Liu, Yang Luh, Yunho Maeng, Manoel Marques, Francisco Jose Martín-Fernández, Douglas T. McClure, David McKay, Srujan Meesala, Antonio Mezzacapo, Nikolaj Moll, Diego Moreda Rodríguez, Giacomo Nannicini, Paul Nation, Pauline Ollitrault, Lee James O’Riordan, Hanhee Paik, Jesús Pérez, Anna Phan, Marco Pistoia, Viktor Prutyanov, Max Reuter, Julia Rice, Abdón Rodríguez Davila, Raymond Harry Putra Rudy, Mingi Ryu, Ninad Sathaye, Chris Schnabel, Eddie Schoute, Kanav Setia, Yunong Shi, Adenilton Silva, Yukio Siraichi, Seyon Sivarajah, John A. Smolin, Mathias Soeken, Hitomi Takahashi, Ivano Tavernelli, Charles Taylor, Pete Taylor, Kenso Trabing, Matthew Treinish, Wes Turner, Desiree Vogt-Lee, Christophe Vuillot, Jonathan A. Wildstrom, Jessica Wilson, Erick Winston, Christopher Wood, Stephen Wood, Stefan Wörner, Ismail Yunus Akhalwaya, and Christa Zoufal, “Qiskit: An open-source framework for quantum computing,” (2019).
- [35] Vlatko Vedral, Adriano Barenco, and Artur Ekert, “Quantum networks for elementary arithmetic operations,” *Phys. Rev. A* **54**, 147–153 (1996).
- [36] Thomas G Draper, “Addition on a Quantum Computer,” (2000), [arXiv:0008033](#).
- [37] Thomas G Draper, Samuel A Kutin, Eric M Rains, and Krysta M Svore, “A logarithmic-depth quantum carry-lookahead adder,” *Quantum Information and Computation* **6**, 351–369 (2006).
- [38] Ghasem Tarmast, “Multivariate Log-Normal Distribution,” *International Statistical Institute Proceedings: 53rd Session, Seoul* (2001).
- [39] Raban Iten, Oliver Reardon-Smith, Luca Mondada, Ethan Redmond, Ravjot Singh Kohli, and Roger Colbeck, “Introduction to universalqcompiler,” (2019), [arXiv:1904.01072](#).
- [40] Yohichi Suzuki, Shumpei Uno, Rudy Raymond, Tomoki Tanaka, Tamiya Onodera, and Naoki Yamamoto, “Amplitude estimation without phase estimation,” (2019), [arXiv:1904.10246](#).
- [41] Mark Broadie and Paul Glasserman, “Estimating security price derivatives using simulation,” *Management Science* **42** (1996).
- [42] Austin G. Fowler, Matteo Mariantoni, John M. Martinis, and Andrew N. Cleland, “Surface codes: Towards practical large-scale quantum computation,” *Phys. Rev. A* **86**, 032324 (2012).
- [43] Miroslav Dobšiček, Göran Johansson, Vitaly Shumeiko, and Göran Wendin, “Arbitrary accuracy iterative quantum phase estimation algorithm using a single ancillary qubit: A two-qubit benchmark,” *Phys. Rev. A* **76**, 030306 (2007).
- [44] C. J. O’Loan, “Iterative phase estimation,” *Journal of Physics A: Mathematical and Theoretical* **43** (2010).
- [45] Krysta M Svore, Matthew B Hastings, and Michael Freedman, “Faster phase estimation,” *Quantum Information & Computation* **14**, 306–328 (2014).