arXiv:2105.12087v1 [quant-ph] 25 May 2021

Quantum algorithm for credit valuation adjustments

Javier Alcazar,[1] Andrea Cadarso,[2] Amara Katabarwa,[1] Marta Mauri,[1] Borja Peropadre,[1] Guoming Wang,[1] and Yudong Cao[1,*]

[1] *Zapata Computing, Inc.*

[2] *BBVA Corporate & Investment Banking, Calle Sauceda 28, 28050 Madrid, Spain*

Quantum mechanics is well known to accelerate statistical sampling processes over classical techniques. In quantitative finance, statistical samplings arise broadly in many use cases. Here we focus on a particular one of such use cases, credit valuation adjustment (CVA), and identify opportunities and challenges towards quantum advantage for practical instances. To improve the depths of quantum circuits for solving such problem, we draw on various heuristics that indicate the potential for significant improvement over well-known techniques such as reversible logical circuit synthesis. In minimizing the resource requirements for amplitude amplification while maximizing the speedup gained from the quantum coherence of a noisy device, we adopt a recently developed Bayesian variant of quantum amplitude estimation using engineered likelihood functions (ELF). We perform numerical analyses to characterize the prospect of quantum speedup in concrete CVA instances over classical Monte Carlo simulations.

I. INTRODUCTION

Statistical simulation tasks are often the most computationally expensive exercises that banks perform. One important class of such exercises is counterparty risk analysis, which has gained increasing importance in the recent years since the Great Recession in the late 2000s. In the aforementioned financial crisis, banks lost tremendous amount of capital in counterparty credit default during derivative transactions, which led to specific regulations and capital requirements. Therefore, risk analyses need to be in place to calculate the precise amounts by which the prices of the derivatives should be adjusted to hedge against the risk of counterparty default, and fall under the general term of credit valuation adjustments (CVA). In response to the credit losses during the Great Recession, the Basel Committee for Banking Supervision (BCBS) has defined regulatory requirements for CVA calculations [1, 50.31-50.36]. The regulation demands that CVA be calculated by simulating the stochastic paths of the underlying exposures, which is frequently carried out by Monte Carlo methods. The probabilistic nature of this process means there will be inherent statistical errors in the resulting CVA estimation. In order to suppress such statistical errors, one must increase the number of stochastic paths sampled. Therefore, a typical CVA calculation for a derivative product involves calculating statistical averages over a large number of price trajectories of the underlying asset(s) of the derivative as well as possible default scenarios of the counterparty. A rough estimation [2] shows that a large number of Monte Carlo samples is needed for meeting the standards laid out in Basel Committee on Banking Supervision's July 2015 consultation document regarding CVA calculations.

The stochastic simulation of the prices of the underlying assets as they change over time is one of the key ingredients of most CVA calculations, except for cases where the expected exposure can be computed analytically. The price fluctuation is typically described as a stochastic process, which is defined for every point in time. Simulating such a stochastic process on modern digital computers introduces a discretized time grid, which introduces a discretization error ϵ_D. Each simulation generates a concrete path of how the price(s) of the asset(s) change(s) over time, which supplies one sample in the Monte Carlo simulation for estimating the expected payoff of the financial instrument that is based on the asset(s). The general goal of the Monte Carlo simulations in financial use cases such as CVA can be described as estimating the expectation value $\mathbb{E}[f(S)]$ of some function f of a set of random variables S. Since one is only able to draw a finite amount of samples or paths on a computer, there is a statistical error ϵ_S associated with each simulation. To estimate the expectation within additive error ϵ_S, generally $\mathcal{O}(1/\epsilon_S^2)$ samples are required. On a classical computer, discretization error introduces additional cost in the simulation. For a discrete approximation scheme of order r on a grid of time interval Δt, the discretization error is $\epsilon_D = \mathcal{O}(\Delta t^r)$ [3]. This gives rise to an extra factor of $\mathcal{O}(1/\epsilon_D^{1/r})$ in the cost of classical simulation. However, such overhead factor can in practice be either mitigated by adopting higher order approximation schemes or multi-level schemes [4]. The cost scaling of $\mathcal{O}(1/\epsilon_S^2)$ with respect to statistical error ϵ_S is a more fundamental artifact of statistics that is in general hard to overcome classically.

The advent of quantum computing presents an exciting opportunity to break the barrier of the classical cost scaling with respect to statistical error. A typical strategy for addressing the problem of estimating $\mathbb{E}[f(S)]$ is by casting it as a problem of estimating an operator expectation: $\mathbb{E}[f(S)] = c\langle\psi|O|\psi\rangle$ for some constant c, state $|\psi\rangle$ and operator O. The problem of estimating $\langle\psi|O|\psi\rangle$ can be addressed by quantum amplitude estimation, with which early proposals [5, 6] demonstrated that one could improve the fundamental cost scaling of parameter estimation from $\mathcal{O}(1/\epsilon_S^2)$ to $\mathcal{O}(1/\epsilon_S)$. This is a significant improvement for applications requiring some

* yudong@zapatacomputing.com

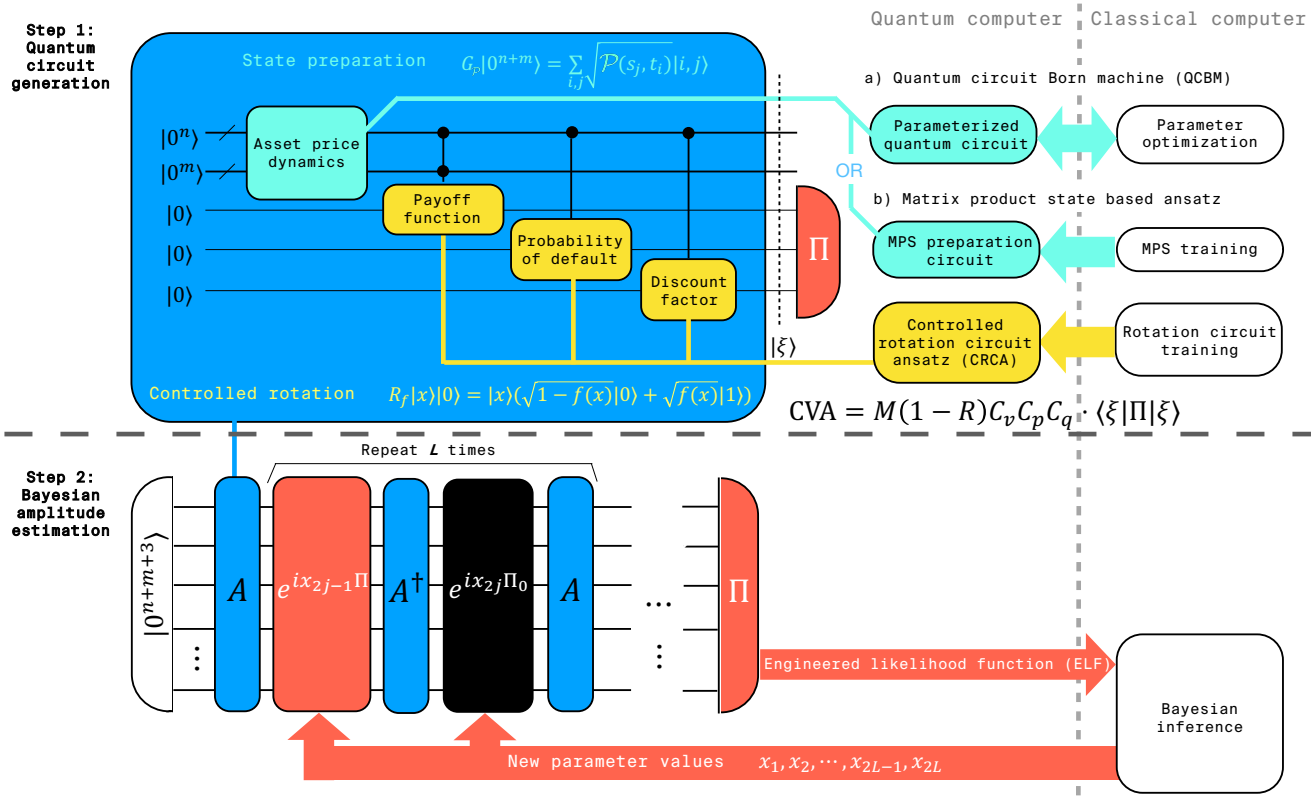


FIG. 1: Overview of the quantum approach to credit valuation adjustment proposed in this work. We start by inspecting the components that make up the CVA quantity (Section II), which translate to the structure of the quantum circuit A in Step 1. It consists of state preparation and controlled rotations (Section III). We expand on the quantum circuit construction in Section IV, where we investigate two alternatives to the state preparation subroutine: quantum circuit Born machine (QCBM) in Section IV.1.1 and matrix product states (MPS) in section IV.1.2. For controlled rotation, we propose controlled rotation circuit ansatz (CRCA) in Section IV.2. For both MPS and CRCA, training occurs only on the classical computer and the resulting parameters are used for constructing the quantum circuit. For QCBM, one trains the quantum circuit iteratively between the quantum and classical computer. At this point the problem of estimating CVA is reduced to estimating the expectation of an observable Π with respect to the output state $|\xi\rangle = A|0^{n+m+3}\rangle$ of the quantum circuit A. We then move on to Step 2 to perform the amplitude estimation using engineered likelihood functions (Appendix E).

control over the statistical error. However, the quantum advantage is realized at a cost of running deep circuits of depth $\mathcal{O}(1/\epsilon_S)$ on a quantum computer. This renders the early algorithms infeasible for near-term quantum devices, which can execute circuits of only finite depth. Recently, there have been various proposals [7–10] for realizing amplitude estimation with reduced depth circuits, at a cost of less quantum speedup compared with the quadratic advantage given by fault-tolerant quantum computers. A general goal of these proposals is to derive as much asymptotic speedup as possible by using the quantum resource available on a given quantum device, even though the speedup may be less than quadratic in many cases.

In this study, we adopt the framework of engineered likelihood function (ELF) proposed in [8, 11] for carrying out the CVA calculation on a quantum computer. This is the first proposal of a quantum algorithm able to tackle the CVA problem, with in-depth discussions about concrete implementations in terms of elementary operations that can be carried out on quantum computers. The same approach can be extended to other risk analysis use cases in quantitative finance. A particular advantage of the ELF framework is that it does not impose *a priori* an amount of quantum resource required for carrying out a certain task, but instead it adaptively takes advantage of however much quantum resource one can afford on a noisy quantum device. To illustrate the concreteness we are able to associate to our solution of the CVA problem, our numerical results indicate that on a quantum computer that has all-to-all qubit connectivity, physical gate error rate 10^{-3}, and uses the surface code of distance 18 for error correction (with cycle time $1\mu s$), performing CVA calculation under the specification listed in Table I within relative error of 0.001% takes around 4.9×10^5 seconds, while the same calculation takes around 3.7×10^6 seconds on a single-core classical computer (4GB RAM, up to 3.1GHz). Such comparison is certainly subject to changes in the details of both the classical and quantum implementations. However, the level of specificity at which we are able to carry out the resource estimation makes it straightforward to account for such implementa-

tion details if necessary. With the engineered likelihood function (ELF) technique developed previously [8, 11] we are able to produce concrete estimates of quantum runtime as the noise and error parameters of the hardware improve.

Our overall approach for the quantum calculation of CVA can be divided into two steps (Figure 1): quantum circuit generation and amplitude estimation. The latter has been discussed in the preceding paragraphs. The former can be further divided into state preparation and controlled rotation subroutines. For both subroutines we have identified opportunities to drastically reduce the depths of quantum circuits compared with those produced from reversible logic synthesis (RLS) [12–15].

Related works. There are many applications in quantitative finance such as derivative pricing and risk analysis that amount to performing integration on domains of stochastic variables. During the early days of quantum computing [16–18], it has been recognized that, for those integration tasks, one is able to glean a quadratic speedup in cost scaling with respect to the statistical error ϵ_S by using insights on quantum counting [19], which later on was developed into quantum amplitude estimation [5]. This line of inquiry was later extended to concrete quantitative finance use cases such as option pricing [20, 21] and risk analysis [22]. However, no use cases related to valuation adjustment (XVA) have been considered so far, making our proposal the first of such studies. There has been also a line of research [23, 24] focusing on using quantum computers for accelerating Monte Carlo calculations; these studies are based on the availability of quantum oracles that can implement certain functions, without considering how these oracles are implemented. Here in our work, however, we will discuss the detailed implementation of the oracles relevant for the CVA problem using elementary quantum operations. Our broader motivation is to make the quantum algorithm as concrete as possible so that it will be amenable for comparison with existing classical solutions as well as implementation on near-term quantum devices.

II. CREDIT VALUATION ADJUSTMENT

Financial derivatives are essentially contracts between two parties. For example, an option contract is a guarantee that one can buy or sell a set of underlying assets at a particular price before or on a specific date (depending on the type of option contract). However, it is possible that when the option contract is exercised, namely when one of the parties decides to go through with the transaction (buying or selling), the counterparty is not able to honor the contract, e.g., the party responsible for buying the asset(s) does not have enough capital to make the purchase. This is a default event that leads to a loss on the selling party. Since an option contract offers insurance against price fluctuation of the underlying asset(s), it has an intrinsic value for which one must pay a premium in order to enter it. In the event of default, the premium paid by the party entering the option contract is essentially lost. Therefore, if there is risk that such default may happen, the premium to be paid for the option contract should be accounted for in its price. The fair amount with which one should make the discount is the value of CVA.

In this study, we focus on CVA problems for European Call options. In general, the CVA quantity is built from the following components:

1. The probability distribution of asset prices $P(s|t)$ at time t. Classically, for a given future time $t = \tau$, sampling from $P(s|t = \tau)$ is typically achieved by performing stochastic simulation of how the asset price s fluctuates as a function of time up to τ, and the set of final price values is the set of samples from $P(s|t = \tau)$. One of the most common stochastic processes used for modeling price fluctuation is geometric Brownian motion.

2. The net amount $v(s, t)$ gained by the purchaser of the option contract, or *payoff*, at asset price s and time t. For a given future time $t = \tau$, the *expected exposure* $E(\tau) = \mathbb{E}_{P(s|\tau)}[v(s, \tau)]$ characterizes the average worth of the option at time τ. Classically, this is estimated by averaging over the values of v computed for each of the samples generated in the previous step. In particular, for the case of European options the payoff is the maximum value between zero and the difference between the price of the asset at maturity and a fixed price K predetermined at the start of the contract, the *strike price*.

3. The probability of default $q(t)$ at time t. One method for modeling $q(t)$ is to consider it as a Poisson process where the Poisson parameter is time dependent. Its exact time dependence can be bootstrapped efficiently, and calibrated from market quantities such as CDS spreads [25].

4. The discount factor $p(t)$. It expresses the time value of money, and it is used to determine the present value of an asset in the future. The formula for the discount factor will depend on the number of periods considered for interest rate payments, where a typical choice is as continuous compound interest, which corresponds to discount factor $p(t) = e^{-rt}$ for an interest rate r. The interest rate can also be time dependent.

The CVA is then calculated as the expected value under the probability measure $q(t)$ of the capital at risk, i.e., how much can be lost if the counterparty does not honor its part in the contract, which in our case of study corresponds to a positive payoff, discounted to the present

and corrected by a loss given default (LGD) factor $1-R$:

$$\begin{aligned}\text{CVA} &= (1-R)\cdot\mathbb{E}_{q(t)}[p(t)E(t)]\\ &= (1-R)\cdot\mathbb{E}_{q(t)}\{p(t)\mathbb{E}_{P(s|t)}[v(s,t)]\}\\ &= (1-R)\int_0^T q(t)p(t)\int_0^\infty P(s|t)v(s,t)\mathrm{d}s\mathrm{d}t.\end{aligned} \tag{1}$$

Here R is the *recovery rate*, defined as the fraction of the value of an asset retained after the borrower defaults. The above CVA calculation can also be generalized to the case where the option contract has multiple underlying assets. In both the single-asset and the multi-asset cases, estimating the value of CVA within statistical error ϵ_S costs $\mathcal{O}(1/\epsilon_S^2)$.

From Equation (1) one sees that the CVA value is an integral over time and price space. Hence, a starting point for estimating the CVA is to approximate it by a sum over its value over discretized time steps $\{t_i\}_{i=0}^M$ where $t_0=0$ and $t_M=T$ (note that the summation starts from $i=1$ while the definitions of t_i start from $i=0$):

$$(1-R)\cdot\sum_{i=1}^{M}E(t_i)p(t_i)q(t_i), \tag{2}$$

where $p(t_i)=\exp(-r_{t_i}t_i)$ is the risk-free discount factor with time-dependent interest rate r_{t_i} at time t_i, $q(t_i)$ is the probability of default between time t_{i-1} and t_i. Moreover, $E(t_i)=\mathbb{E}_{P(s|t_i)}[v(s,t_i)]$ is the expected exposure with

$$v(s,t)=\max\{s(t)-K\exp(-r_t(T-t)),0\}, \tag{3}$$

being the payoff at maturity time T for a strike price K, assuming a single underlying asset for the European option. Here, $s(t)=s(0)\exp\left(\sigma\xi+\left(\mu-\frac{\sigma^2}{2}\right)t\right)$ is the asset price at time t, modeled as a geometric Brownian motion where $\xi\sim N(0,1)$ is a unit normal random variable, σ is the volatility of the asset and μ the market drift, accounting for the long-term price movement trend on average at the risk-free rate.

To enable representation of the asset price using quantum registers, we also discretize the value of the price with $N+1$ values $\{s_j\}_{j=0}^N$. We can then approximate the distribution of the asset price fluctuation by considering the joint distribution $P(s,t)=P(s|t)P(t)$ where the marginal distribution $P(t)$ is the uniform distribution over the time period from 0 to T. We then define a discrete probability distribution $\mathcal{P}$ defined at each point (s_j,t_i) for approximating $P(s,t)$:

$$\mathcal{P}(s_j,t_i)=\frac{1}{M\mathcal{N}}\int_{s_{j-1}}^{s_j}P(s|t_i)\mathrm{d}s. \tag{4}$$

where $i=1,\cdots,N$, $j=1,\cdots,M$ and $\mathcal{N}=\int_{s_0}^{s_N}P(s,t_i)\mathrm{d}s$ is the normalization constant. Since the marginal distribution $P(t)$ is uniform, after discretization the marginal distribution of $\mathcal{P}$ should satisfy $\mathcal{P}(t_i)=1/M$. This produces an approximation of the expected exposure as

$$\begin{aligned}\tilde{E}(t_i) &= \sum_{j=1}^{N}\frac{\mathcal{P}(s_j,t_i)}{\mathcal{P}(t_i)}v(s_j,t_i)\\ &= M\sum_{j=1}^{N}\mathcal{P}(s_j,t_i)v(s_j,t_i)\end{aligned} \tag{5}$$

Combining the discretizations for both asset price and time, we arrive at the quantity to be estimated as

$$M(1-R)\cdot\sum_{i=1}^{M}\sum_{j=1}^{N}\mathcal{P}(s_j,t_i)v(s_j,t_i)p(t_i)q(t_i). \tag{6}$$

Note that in Equation (6) the quantities $\mathcal{P}$, p and q are bounded between 0 and 1, while the payoff function v may not be so. Since the discretization in asset price s means the value of s is bounded, the value of v must also be bounded. We introduce a scaling factor C_v such that $v=C_v\tilde{v}$ where $\tilde{v}$ is bounded between 0 and 1. For quantities p and q, it is possible that their values vary only subtly over their entire domains, making it hard to accurately approximate them. We therefore introduce scaling factors C_p and C_q such that $p=C_p\tilde{p}$ and $q=C_q\tilde{q}$. By letting $C_p>1$ and $C_q>1$ we are able to amplify the fluctuations of the functions p and q on their domains respectively. This leads to the final expression

$$\begin{aligned}\widetilde{\text{CVA}} &= M(1-R)C_vC_pC_q\cdot\\ &\quad\underbrace{\sum_{i=1}^{M}\sum_{j=1}^{N}\mathcal{P}(s_j,t_i)\tilde{v}(s_j,t_i)\tilde{p}(t_i)\tilde{q}(t_i)}.\end{aligned} \tag{7}$$

The problem then becomes casting the bracketed term in Equation (7), which is bounded between 0 and 1, as an amplitude estimation problem.

CVA instance for benchmarking. We consider a specific instance of the CVA problem defined on a single asset European call option: all the calculations were carried out under the specification listed in Table I. The time points $\{t_i\}_{i=1}^M$ used for this instance are generated using the formula

$$t_i=i\cdot\frac{T}{M},\quad i=1,\cdots,M \tag{8}$$

where $M=2^m$ is chosen such that the time points can be represented by the computational basis state of an m-qubit register. The maturity time of 6 months corresponds to $T=\frac{184}{360}\approx 0.511$, namely the number of business days (184) in the 6-month duration starting from March 5th, 2020, divided by the total number of days considered in a year under Day Count convention (360). Using the *Actual* convention for day count means that all days between two dates are considered for interest accrual. If a given date is not a business day, it is adjusted according to the *Following* convention, which considers the first business day after the given holiday. The

method used to determine the day at which payments are due is the *IMM* convention (International Money Market month), namely the effective dates are taken to be the third Wednesday of March, June, September and December. The *quarterly* frequency indicates how often payments are due. In order to compute default probabilities, we use a bootstrapping approach to recover hazard rates from market CDS quote spreads [25], where the interest rate curve is variable over time, specifically, it is taken to be the EONIA curve.

Initial Asset Price	5.0
Strike Price	5.5
Volatility	0.25
Drift	0.02
Maturity	6 months
Start Date	05/03/2020
CVA Recovery Rate	0.415
Notional	1
Forward Rate Curve	EONIA Curve
Hazard Rates	Flat Piecewise
CDS Quote Spreads	[0.00093772, 0.00184451, 0.0032286, 0.0047065, 0.00574888, 0.00574888]
CDS Tenors	[1y, 3y, 5y, 7y, 10y, 15y]
CDS Recovery Rate	0.4125
CDS Settlement Days	0
Calendar	Target
Day Count	Actual / 360
Business Day Convention	Following
Date Generation	IMM
Frequency	Quarterly

TABLE I: Specification of the CVA instance benchmarked in this study.

Benchmark value using Monte Carlo simulation. As a classical benchmark for numerically testing the quantum algorithm, we ran Monte Carlo simulations that use 10^5 stochastic paths mimicking the asset price fluctuation over time. The simulation results allow us to estimate the expected exposure $E(t)$ and approximate CVA by Equation (2) directly, without applying the price discretization that produces $\mathcal{P}$. The Monte Carlo simulations show that the CVA value for this particular instance is

$$\mathrm{CVA}_{\mathrm{MC}} = (5.599 \pm 0.002) \cdot 10^{-5}. \tag{9}$$

The calculation is implemented using the Orquestra® integration with the `quantlib` library [26]. For the remainder of the paper, we will use the value of $\mathrm{CVA}_{\mathrm{MC}}$ as the benchmark value representing the "exact" value of CVA, with which CVA calculations by other methods in this study are compared.

Benchmark value with price discretization. The value of CVA in Equation (9) assumes discretization in time according to Equation (8) but the price can take any value from 0 to infinity. To prepare for quantum algorithm treatment, we also discretize the price, leading to the discrete distribution $\mathcal{P}$ in Equation (4). In Section IV.1 we provide more details on the construction of $\mathcal{P}$.

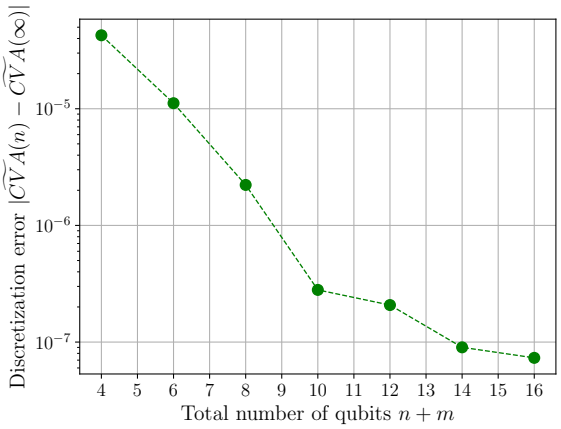


FIG. 2: The convergence of $\widetilde{\mathrm{CVA}}(n)$ as n grows. Here the total number of qubits is $n+m$, where the number of qubits m for representing time (Equation 8) is fixed to be 2.

Clearly, such CVA estimation based on discretized price values depends on the number N of discrete price values. Let $\widetilde{\mathrm{CVA}}(n)$ be the value obtained using $N = 2^n$ discrete price values evenly spaced between s_1 and s_N. Here we choose N values that are powers of two for its convenience in associating with the number of qubits $n = \log_2 N$ needed for encoding each price value s_j. Numerical results (Figure 2) indicate that as n grows, $\widetilde{\mathrm{CVA}}(n)$ converges towards a value

$$\widetilde{\mathrm{CVA}}(\infty) = (5.48 \pm 0.02) \cdot 10^{-5}. \tag{10}$$

The difference between $\widetilde{\mathrm{CVA}}(\infty)$ and $\mathrm{CVA}_{\mathrm{MC}}$ is likely due to the probability weight lost when restricting the asset price domain from $[0, \infty)$ to $[s_1, s_N]$. However, this difference accounts for only around 2% relative error with respect to $\mathrm{CVA}_{\mathrm{MC}}$. As shown in Figure 2, for small values of n, the error due to discretization (namely the introduction of $\{(s_j, t_i) | j = 1, \cdots, N; i = 1, \cdots, M\}$ for representing the domain of time and price) is dominated by the number of discrete values s_j that represent the price.

III. QUANTUM ALGORITHM

In Section II, the CVA calculation is broken down into 4 components. To describe how the quantum algorithm works, we group the 4 steps into two parts: step 1 being the state preparation, steps 2-4 being the controlled rotation implementation. The final step is then assembling the operations from the previous 4 steps into a quantum

circuit whose output state allows for the measurement of the bracketed term in Equation (7).

State preparation. The goal here is to (approximately) realize an operation $G_{\mathcal{P}}$ acting on $n+m$ qubits, where $n = \lceil \log_2 N \rceil$ and $m = \lceil \log_2 M \rceil$, such that

$$G_{\mathcal{P}} \left|0^{n+m}\right\rangle = \sum_{i,j} \sqrt{\mathcal{P}(s_j, t_i)} \left|i, j\right\rangle . \tag{11}$$

Here $|i\rangle$ (resp. $|j\rangle$) is the computational basis state marking the index i (resp. j) in its binary form. The setting of the marginal distribution of $\mathcal{P}$ over the asset prices is a log-normal distribution, as a consequence of choosing the geometric Brownian model as the statistical model for the underlying asset. A different distribution may be chosen for other statistical models. For example, in cases where one would like to contemplate distributions with heavy tails, Lévy distribution may be used for the marginal distribution over the asset prices.

Controlled rotation. The goal here is to use a quantum state $|x\rangle$ encoding an input variable x as a control register for enacting a rotation on an ancilla qubits that results in a state $\sqrt{1-f(x)}|0\rangle + \sqrt{f(x)}|1\rangle$ for some function f. This construction is also commonly used in previous works [23, 27] on quantum speedups for Monte Carlo procedures. For our purpose, we introduce a controlled rotation operator for each of the steps 2-4 described in Section II.

For representing the payoff of the option contract, we define quantum operator R_v acting on $n+m+1$ qubits such that

$$R_v \left|i\right\rangle \left|j\right\rangle \left|0\right\rangle = \left|i\right\rangle \left|j\right\rangle \left(\sqrt{1-\tilde{v}(s_j,t_i)}\left|0\right\rangle + \sqrt{\tilde{v}(s_j,t_i)}\left|1\right\rangle\right). \tag{12}$$

For representing the probability of default, we introduce operator R_q acting on $n+1$ qubits be such that

$$R_q \left|i\right\rangle \left|0\right\rangle = \left|i\right\rangle \left(\sqrt{1-\tilde{q}(t_i)}\left|0\right\rangle + \sqrt{\tilde{q}(t_i)}\left|1\right\rangle\right). \tag{13}$$

For representing the discount factor, we define quantum operator R_p acting on $n+1$ qubits such that

$$R_p \left|i\right\rangle \left|0\right\rangle = \left|i\right\rangle \left(\sqrt{1-\tilde{p}(t_i)}\left|0\right\rangle + \sqrt{\tilde{p}(t_i)}\left|1\right\rangle\right). \tag{14}$$

Quantum circuit assembly. We could then describe a procedure for estimating the bracketed quantity in (7) as the following (Figure 3):

1. Start with two quantum registers, one of n qubits and the other of m qubits and generate the quantum state

$$\sum_{i=1}^{M}\sum_{j=1}^{N} \sqrt{\mathcal{P}(s_j,t_i)}\left|i\right\rangle\left|j\right\rangle . \tag{15}$$

using the operator $G_{\mathcal{P}}$.

2. Add an ancilla qubit in $|0\rangle$ for storing the payoff function ($p.f.$) and apply the operator R_v to produce an entangled state

$$\sum_{i=1}^{M}\sum_{j=1}^{N} \sqrt{\mathcal{P}(s_j,t_i)}\left|i\right\rangle\left|j\right\rangle \otimes \left(\sqrt{1-\tilde{v}(s_j,t_i)}\left|0\right\rangle_{p.f.} + \sqrt{\tilde{v}(s_j,t_i)}\left|1\right\rangle_{p.f.}\right). \tag{16}$$

3. Add another ancilla qubit in $|0\rangle$ for storing the probability of default ($p.o.d.$) and apply the operator R_q onto the first register and the new ancilla qubit to produce the state

$$\sum_{i=1}^{M}\sum_{j=1}^{N} \sqrt{\mathcal{P}(s_j,t_i)}\left|i\right\rangle\left|j\right\rangle \otimes \left(\sqrt{1-\tilde{v}(s_j,t_i)}\left|0\right\rangle_{p.f.} + \sqrt{\tilde{v}(s_j,t_i)}\left|1\right\rangle_{p.f.}\right) \otimes \left(\sqrt{1-\tilde{q}(t_i)}\left|0\right\rangle_{p.o.d.} + \sqrt{\tilde{q}(t_i)}\left|1\right\rangle_{p.o.d.}\right). \tag{17}$$

4. Add another ancilla qubit in $|0\rangle$ for storing the discount factor ($d.f.$) and apply the operator R_p onto the first register and the new ancilla qubit to produce the final state

$$\begin{aligned}\left|\xi\right\rangle = \sum_{i=1}^{M}\sum_{j=1}^{N} &\sqrt{\mathcal{P}(s_j,t_i)}\left|i\right\rangle\left|j\right\rangle \otimes \left(\sqrt{1-\tilde{v}(s_j,t_i)}\left|0\right\rangle_{p.f.} + \sqrt{\tilde{v}(s_j,t_i)}\left|1\right\rangle_{p.f.}\right) \otimes \\ &\left(\sqrt{1-\tilde{q}(t_i)}\left|0\right\rangle_{p.o.d} + \sqrt{\tilde{q}(t_i)}\left|1\right\rangle_{p.o.d}\right) \otimes \\ &\left(\sqrt{1-\tilde{p}(t_i)}\left|0\right\rangle_{d.f.} + \sqrt{\tilde{p}(t_i)}\left|1\right\rangle_{d.f.}\right)\end{aligned} \tag{18}$$

5. Let Π be projector onto the subspace where the $d.f.$, $p.f.$ and $p.o.d.$ ancilla qubits are all in the state $|1\rangle$. More explicitly, we have

$$\begin{aligned}\Pi &= \left|1\right\rangle\left\langle 1\right|_{d.f.} \otimes \left|1\right\rangle\left\langle 1\right|_{p.f.} \otimes \left|1\right\rangle\left\langle 1\right|_{p.o.d.} \\ &= \frac{1}{8}(I - Z_{d.f.} - Z_{p.f.} - Z_{p.o.d.} \\ &\quad + Z_{d.f.}Z_{p.f.} + Z_{d.f.}Z_{p.o.d.} + Z_{p.f.}Z_{p.o.d.} \\ &\quad - Z_{d.f.}Z_{p.f.}Z_{p.o.d.}),\end{aligned} \tag{19}$$

which is a linear combination of Pauli operators that can be measured directly and simultaneously on the quantum processor.

We observe that the quantity desired in Equation (7) can be obtained by

$$\sum_{i=1}^{M}\sum_{j=1}^{N} \mathcal{P}(s_j,t_i)\tilde{v}(s_j,t_i)\tilde{p}(t_i)\tilde{q}(t_i) = \left\langle\xi\right|\Pi\left|\xi\right\rangle , \tag{20}$$

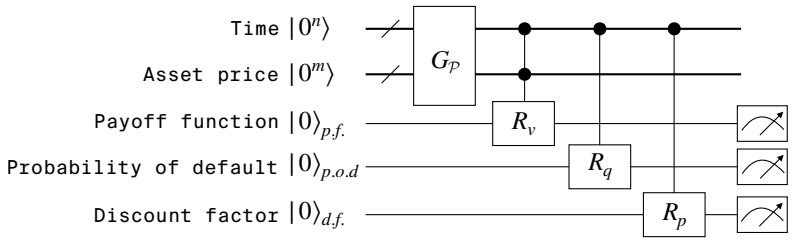


FIG. 3: The proposed (ideal) quantum circuit to perform credit value adjustment as described in Section III.

which can be estimated within a sampling error of ϵ_S in time $\mathcal{O}(1/\epsilon_S)$ using amplitude estimation.

The rest of the paper is organized as follows. In Section IV we describe in details the quantum circuit construction used for preparing the state $|\xi\rangle$, using a particular circuit as an example. The quantum circuit construction involves the following operations:

- Operator $G_{\mathcal{P}}$ for state preparation (Section IV.1);
- Controlled operations R_v, R_q, R_p (Section IV.2).

The proposed quantum circuit for the algorithm can be found in Figure 3. In Appendix E we proceed to describe how the quantum circuit design is used for amplitude estimation by engineered likelihood function (ELF) [8].

IV. QUANTUM CIRCUIT

Existing state preparation techniques [12–15] rely on the ability to perform operations such as evaluating arithmetic expressions [14], computing the integral of a probability density function over an interval [12], and extracting elements of a sparse matrix [13]. These operations are either assumed to be supplied through oracles, in which case their implementations in terms of elementary quantum operations on a quantum computer are entirely not taken in to account, or assumed to be realizable efficiently via well-known techniques such as reversible logic synthesis (RLS) [28–33]. However, although RLS is efficient in an asymptotic sense if the underlying function can be realized by a polynomial-time classical procedure, concrete resource estimations show that it is costly compared with other parts of quantum algorithms, motivating alternative approaches [34]. In this paper we also perform numerical experiments to illustrate how costly RLS can be with even small examples of CVA calculations (Section IV.2).

Instead of RLS, which is capable of enabling both the controlled rotation and the state preparation [35], in this work we identify opportunities for realizing both with circuits of much shorter depths. For state preparation, we investigate two alternatives: quantum circuit Born machine [36] and quantum circuit construction based on matrix product states [37]. We show that highly accurate approximations are possible with circuits that are much shallower than those produced from RLS. We note that there are also other state preparation schemes [38–40] inspired by generative adversarial networks that can also yield more efficient state preparation protocols than RLS.

For the purpose of illustration, we consider a specific instance of CVA estimation using Equation (7) with parameters listed in Table II.

The classical benchmark value (Section II) for this instance is

$$\widetilde{\mathrm{CVA}}(2) = 1.223 \cdot 10^{-5}. \tag{21}$$

The large discrepancy between $\widetilde{\mathrm{CVA}}(2)$ and $\mathrm{CVA}_{\mathrm{MC}}$ in Equation (9) can be mostly attributed to discretization error due to finite n (Figure 2). In Section IV.3 we will compute the value $\widetilde{\mathrm{CVA}}_Q$ produced by the quantum circuit for this instance, compare it with $\widetilde{\mathrm{CVA}}(2)$ and discuss the sources of error.

Number of qubits m for encoding time	2
Number of qubits n for encoding price	2
Scaling constant for payoff C_v	1.8201814
Scaling constant for default probability C_q	0.0002038
Scaling constant for discount factor C_p	1

TABLE II: Parameters of the example quantum circuit considered in Section IV. The other parameters related to the CVA instance are shown in Table I.

IV.1. State preparation

In the context of the CVA problem (Section III), the role of state preparation is to implement the $G_{\mathcal{P}}$ operator, thus preparing a quantum state loading into it the target distribution $p_{tg}(x) = \mathcal{P}(s_j, t_i)$ which is the discretized joint distribution of time and asset price. The circuit acts on two registers of qubits encoding time and asset price respectively, where the qubit states are represented as bitstrings whose first part refers to the time register and the following to the price register. Given that we are considering asset fluctuations modeled by geometric Brownian motion, at each point in time such distribution is the log-normal distribution, namely:

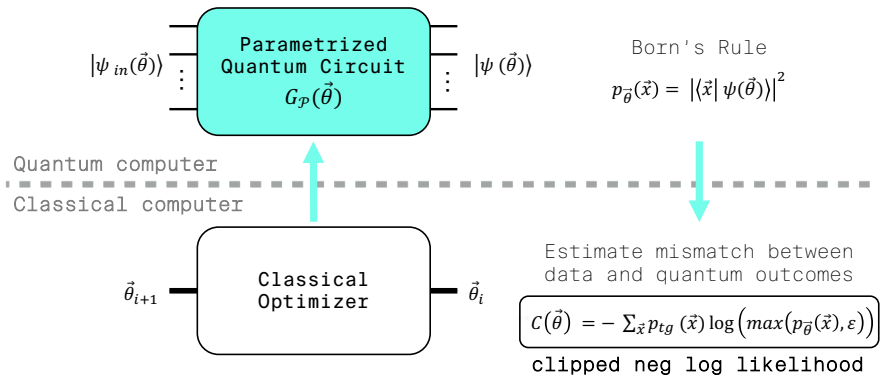


FIG. 4: Quantum circuit Born machine for learning the state preparation circuit $G_{\mathcal{P}}$.

$$P(s|t) = \frac{1}{s\sqrt{2\pi\sigma^2}} \exp\left[-\left(\frac{\ln s - \ln s_0 - (\mu - \frac{\sigma^2}{2})t}{\sqrt{2\sigma^2 t}}\right)^2\right], \tag{22}$$

where σ is the price volatility, μ represents the market drift which accounts for the long term price movement trend, and s_0 is the initial asset price.

The target distribution of $\mathcal{P}$ in Equation (4) is obtained via classical Monte Carlo simulation of the asset prices and then discretized to the available quantum states $|x\rangle = |i, j\rangle$. Specifically, 10^5 trajectories of asset prices dynamics from time 0 to maturity T are computed, simulating geometric Brownian motion on a fine time grid. Then, the asset price distributions at the time steps t_i defined by Equation (8) are extracted from the above simulation: for each time step t_i we obtain a log-normal peak as in Equation (22). We choose the range of the discrete s_j values such that the smallest value $s_1 = \max\{\hat{\mu} - 3\hat{\sigma}, 0\}$ and the largest value $s_N = \hat{\mu} + 3\hat{\sigma}$, where $\hat{\mu}$ and $\hat{\sigma}$ are the sample mean and sample standard deviation of the price values produced by the Monte Carlo simulations. We then calculate $\mathcal{P}$ by binning the price data from the Monte Carlo simulation yielding CVA_{MC} according to the different s_j values, which enabled the estimation of CVA according to Equation (7).

IV.1.1. Quantum Circuit Born Machine

Quantum Circuit Born Machine (QCBM) [36] has been shown to learn and load a target probability distribution p_{tg} into a quantum state. The structure of this hybrid quantum-classical algorithm is depicted in Figure 4. The subroutine running on the quantum computer consists of training a parametrized quantum circuit, depending on some parameters $\vec{\theta}$ and encoding a probability distribution $p_{\vec{\theta}}$. Indeed, the output state of the circuit $|\psi(\vec{\theta})\rangle$ contains in its amplitudes the probability distribution, according to Born's rule:

$$p_{\vec{\theta}}(x) = |\langle x|\psi(\vec{\theta})\rangle|^2, \tag{23}$$

where $|x\rangle = |i, j\rangle$ represents a computational basis state that encodes a point on the discretized domain of the target probability distribution p_{tg}. The circuit parameters $\vec{\theta}$ are tuned in order to find the optimal set $\vec{\theta}^*$ such that $p_{\vec{\theta}}$ is as close as possible to p_{tg}. The process of learning the parameters is carried out by a classical optimizer: its goal is to minimize a cost function which quantifies the difference between the two probability distribution into play. We choose to employ an evolutionary and derivative free strategy as the classical optimizer, namely the Covariance Matrix Adaptation Evolution Strategy (CMA-ES [41]). As for the cost function, we use the clipped negative log-likelihood [36], defined as follows:

$$C(\vec{\theta}) = -\sum_x p_{tg}(x) \log(\max(p_{\vec{\theta}}(x), \epsilon), \tag{24}$$

where ϵ is a small parameter to avoid singularity. There are several options available for both the optimizer and the cost function, whose efficiency is highly dependent on the specific problem instance.

The QCBM is able to learn the desired distribution $\mathcal{P}(s_j, t_i)$, by undergoing a training process that tunes the parameters of the quantum circuit so that the desired target is loaded into a quantum state. The ansatz we consider for the parametrized quantum circuit, defined on 4 qubits, is shown in Figure 5.

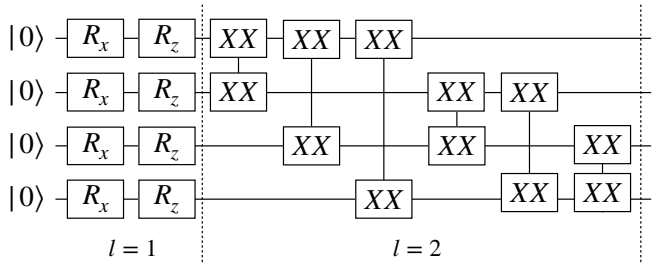


FIG. 5: The ansatz circuit for realizing the operator $G_{\mathcal{P}}$, with 14 variational parameters (one for each gate). It is composed of a single qubit layers ($l = 1$) and an entangling layer ($l = 2$), so that overall it has $n_{\text{layers}} = 2$.

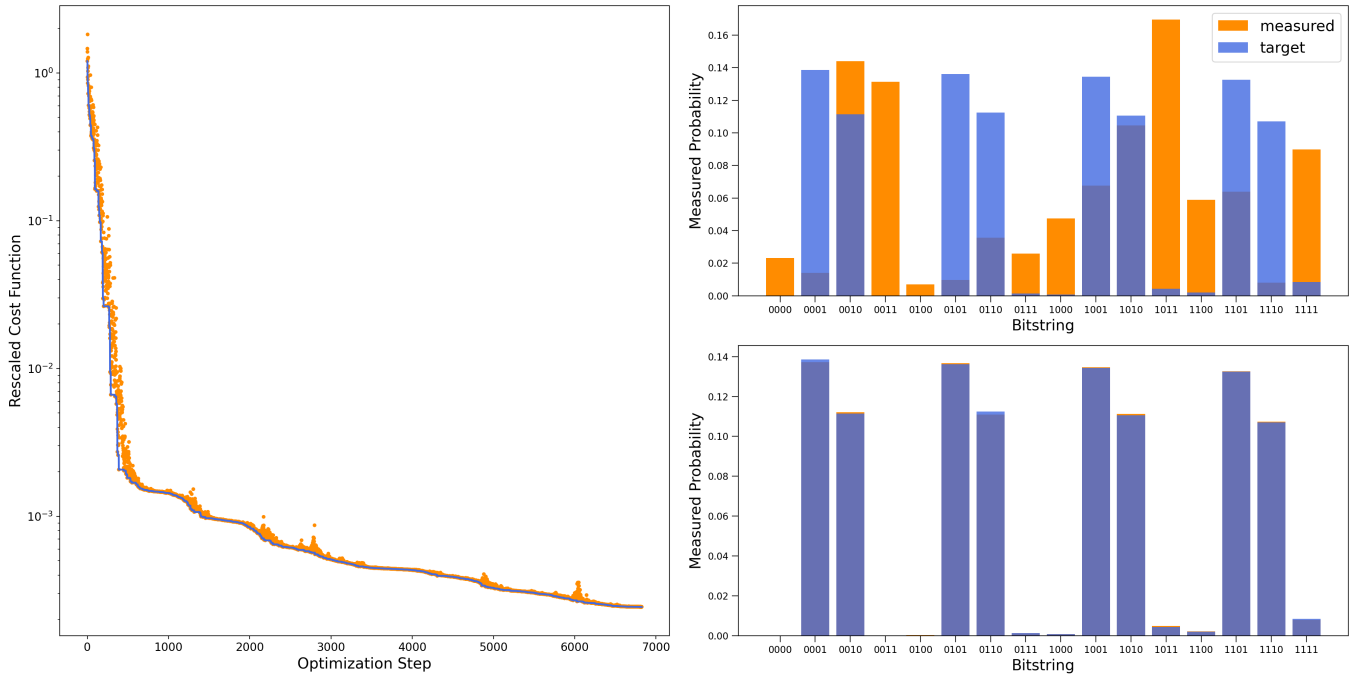


FIG. 6: The training process of the parametrized quantum circuit in the single asset case, where the ansatz is defined on 4 qubits and has 14 gates. On the left, the plot shows the convergence of the (rescaled) cost function towards its ideal value. The orange dots mark the cost function value at each optimization step, whereas the blue line connects only points with progressively decreasing values, thus highlighting the minimization trend. On the right, histograms display the probability distributions encoded in the quantum state before (top) and after (bottom) the training.

It is composed of two layers, where the first layer ($l = 1$) is made of single qubit rotations and the second one ($l = 2$) uses two qubit gates to introduce entanglement. The depth of the circuit can be varied as needed, so that each layer of one(two)-qubit(s) gates is identified by an odd(even) index l, with $l = 1, \cdots, n_{\text{layers}}$. The types of one-qubit gates used in the QCBM circuit are chosen to be X and Z rotations, whereas for the two-qubits gates we use the XX coupling gate, which is implemented natively in ion-trap quantum computers. Specifically, the gates are defined as follows:

$$\begin{aligned} R_{x,z}(\theta) &= \exp\left(-i\tfrac{\theta}{2}\sigma_{x,z}\right) \\ XX(\theta) &= \exp\left(-i\theta\sigma_x \otimes \sigma_x\right). \end{aligned} \tag{25}$$

The qubit connectivity is assumed to be all-to-all, so that each pair of qubits undergoes an XX transformation. The depth of the circuit grows according to the circuit size: the larger the number of qubits, the deeper the circuit needs to be in order to be trained successfully (see Appendix A). For the 4-qubit instance in Figure 5, one layer of tunable entangling gates is enough for the QCBM to be able to learn the target distribution. Figure 6 shows a typical training curve of the cost function, as the iterations proceed. Note that the cost function has been rescaled so that its expected value for a successful training is zero.

IV.1.2. Matrix Product State

An alternative method to using QCBM is to leverage Matrix Product State (MPS) [37], which is a tensor network that has been used for mimicking correlations between different variables. An MPS is represented by a sequence of tensors $A^{(1)}$, $A^{(2)}$, $\cdots$, $A^{(n+m)}$. Each tensor $A^{(\ell)}$ depends on the value of x_ℓ, which is the ℓ-th element of $x = |i, j\rangle$. When evaluated at a particular value of x_ℓ, $A^{(\ell)}$ becomes a matrix (except for when $\ell = 1$ or $\ell = n + m$, where it becomes a vector), denoted as $A^{(\ell)}_{x_\ell}$. The MPS can then be defined as (Figure 7a)

$$\Psi_A(x) = \text{Tr}\left(A^{(1)}_{x_1} A^{(2)}_{x_2} \cdots A^{(n+m)}_{x_{n+m}}\right). \tag{26}$$

The equation above assumes that the $A^{(\ell)}_{x_\ell}$ objects have compatible dimensions so that they can be multiplied together properly. If $A^{(\ell)}_{x_\ell}$ is a matrix of dimension $d_\ell \times d_{\ell+1}$ and $A^{(\ell+1)}_{x_{\ell+1}}$ is a matrix of dimension $d_{\ell+1} \times d_{\ell+2}$, then $d_{\ell+1}$ is the *bond dimension* between $A^{(\ell)}_{x_\ell}$ and $A^{(\ell+1)}_{x_{\ell+1}}$.

An MPS can represent a quantum wavefunction $\Psi_A(x)$ that approximates a target distribution p_{tg} in the same way as QCBM, namely $p_A(x) = |\Psi_A(x)|^2 \approx p_{tg}(x)$. A concrete way to measure the closeness between the MPS output p_A and the target distribution p_{tg} is by computing a negative log-likelihood function over a set S of samples

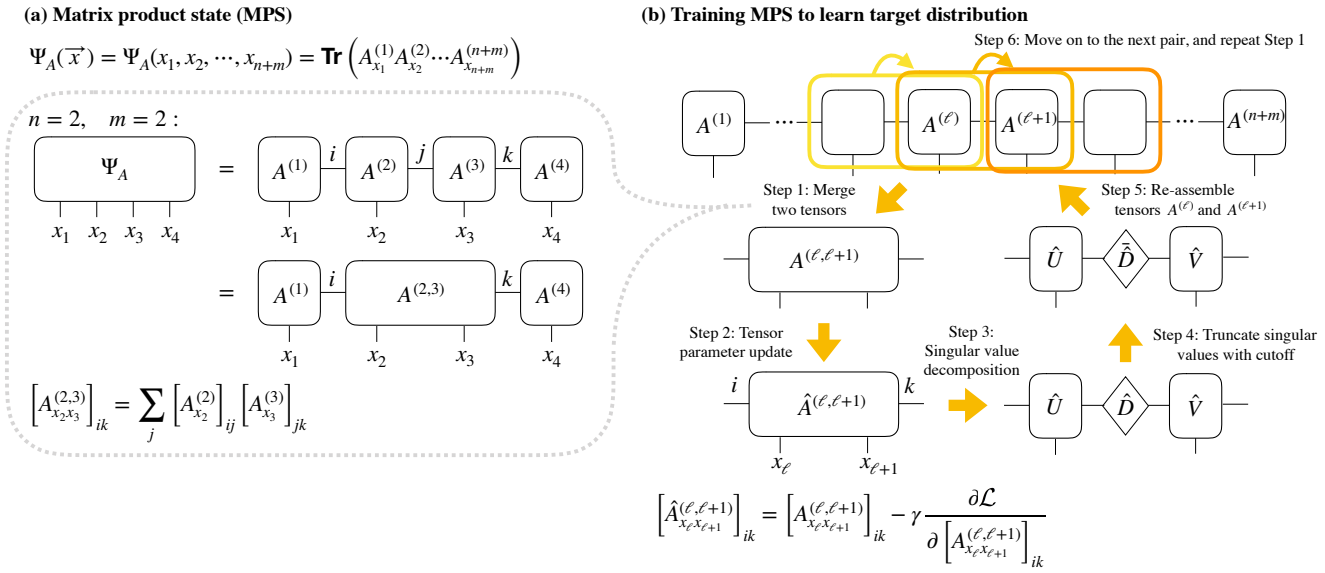


FIG. 7: Matrix product state (MPS) and its usage for learning a target distribution. **(a)** Definition of an MPS $\Psi_A(x)$ and process for contracting two tensors in the MPS. **(b)** Process for training MPS to learn a target distribution. Note that in Step 5 one could either "go left" by letting $A^{(\ell)} = \hat{U} \cdot \tilde{D}$, $A^{(\ell+1)} = \hat{V}$ or "go right" by letting $A^{(\ell)} = \hat{U}$ and $A^{(\ell+1)} = \tilde{D} \cdot \hat{V}$. Details for evaluating the gradient of the cost function $\mathcal{L}$ are shown in [37].

x generated from p_{tg}:

$$\mathcal{L}(A) = -\frac{1}{|S|} \sum_{x \in S} \log p_A(x). \tag{27}$$

The likelihood function is similar to the cost function in Equation 24 in the sense that averaging over S approximates the expectation over the target distribution $-\sum_x p_{tg}(x) \log p_A(x)$.

In order to minimize the negative log-likelihood function, we adopt an iterative scheme. A summary of the scheme is provided in Figure 7b. This method builds on the connection between unsupervised generative modeling and quantum physics, where MPS is employed as a model to learn the probability distribution of a given data set with an algorithm which resembles Density Matrix Renormalization Group (DMRG), which is an efficient algorithm that attempts to find the MPS wavefunction corresponding to the ground state for a given Hamiltonian. Relying on the ability to efficiently evaluate gradients of the objective function [37], we can iteratively improve the MPS approximation of the target distribution. The iterations proceed until one of the three scenarios: 1) a threshold for the KL divergence between the model output and the training data is reached, 2) the difference in the objective function between two adjacent training steps is below a threshold, or 3) a maximum number of iterations is reached. In Figure 8 we show results for training the MPS up to $n_{\text{qubits}} = n + m = 20$ qubits. From the data, the empirical scaling of the MPS training cost is roughly $O(n_{\text{qubits}}^6)$.

In order to make the MPS construction useful for the CVA calculation, as well as to compare QCBM and MPS in a fair way, we need an explicit recipe for generating

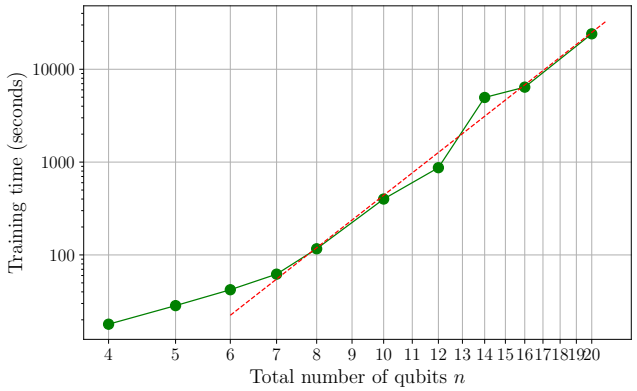


FIG. 8: Runtime for training the MPS versus the number of qubits. The scales for both axes are logarithmic. The red dashed line is a linear regression with correlation coefficient being 0.987 and the slope being roughly 5.827.

quantum circuits that prepare the MPS. The basic idea is to use a sequence of Singular Value Decomposition (SVD) steps to transform the MPS into an orthogonal form in which each tensor is an isometry and hence can be embedded into a unitary operator. Then we decompose these unitary operators into CNOT and single-qubit gates by the method in [42]. When the MPS has n sites and bond dimension D, there are at most $n - \lceil \log_2(D) \rceil$ such unitary operators, where each of them acts on at most $\lceil \log_2(D) \rceil + 1$ qubits and can be implemented with $O(D^2)$ two-qubit gates. As a consequence, the quantum circuit for preparing the MPS contains $O(nD^2)$ two-qubit gates. Note that it remains unknown how many CNOT

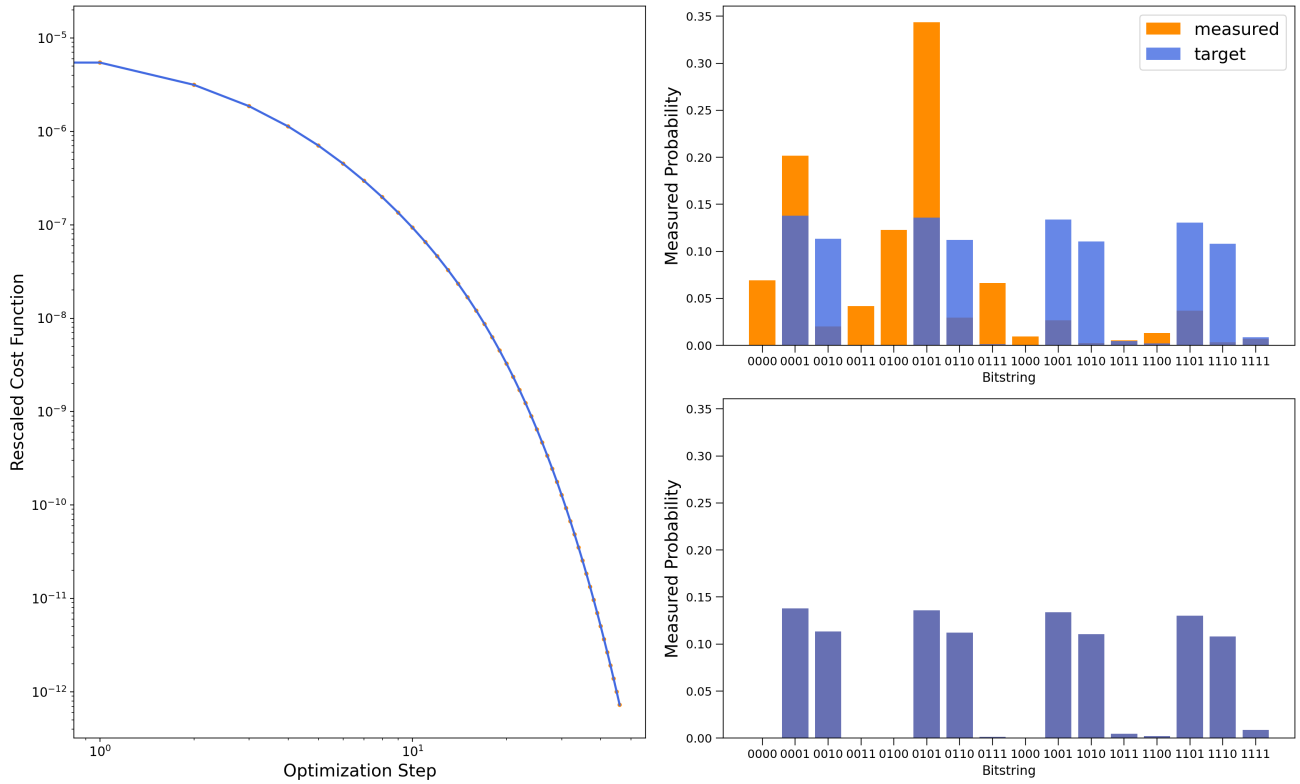


FIG. 9: The training process of the MPS model in the single asset case, where the ansatz is defined on 4 qubits and has 14 gates. On the left, the plot shows the convergence of the cost function towards its ideal value with y-axis rescaled logarithmically. The dots mark the cost function value at each optimization step, whereas the line connects only points with progressively decreasing values, thus highlighting the minimization trend. On the right, histograms display the probability distributions encoded in the quantum state after the first training step (top) and the one after the final step of training (bottom).

n_{qubits}	t_{train}	KL	n_{iters}
4	17.94	$3.21 \cdot 10^{-6}$	2
5	28.44	$4.11 \cdot 10^{-6}$	2
6	42.17	$4.53 \cdot 10^{-6}$	2
7	62.06	$1.62 \cdot 10^{-6}$	3
8	116.46	$8.60 \cdot 10^{-5}$	3
10	399.99	$2.07 \cdot 10^{-2}$	5
12	870.41	$1.5 \cdot 10^{-2}$	7
14	4967.78	$7.04 \cdot 10^{-2}$	19
16	6417.41	0.86	22
20	24084.47	3.20	25

TABLE III: Results for MPS training. Here $n_{\text{qubits}} = n + m$ is the number of qubits encoding the joint time and price values, t_{train} is the total time for training the MPS (in seconds). The maximum bond dimension of the MPS in all cases is limited to 2. The values on this table has been obtained with a threshold for KL divergence of $3 \cdot 10^{-5}$.

and single-qubit gates are needed exactly to implement a general k-qubit unitary operator for $k \geq 3$, and the best known results are lower and upper bounds on them. So we can only give lower and upper bounds on the numbers of elementary gates in the circuit for preparing the MPS (except for the case $D = 2$) in Table IV. See Appendix B for more details about encoding MPS into quantum circuits.

We are now able to compare the two methods proposed so far for the state preparation task. Specifically, we focus on the aforementioned 4-qubits instance of the QCBM with different depths, i.e. $n_{\text{layers}} = [2, 4, 6, 8, 10]$, which corresponds to a 4-site MPS with the same values of bond dimension D. We also consider a 6-qubits QCBM instance and its analogous MPS, varying n_{layers} and D as in the former setting. The target distribution corresponds to Equation (4) with $P(s|t)$ as in Equation (22). Tables V and IV show the results of the comparison between QCBM and MPS-based circuits, where we report the number of one- and two- qubit gates (in terms of CNOTs for a fair comparison), the number of iterations until convergence of the cost function minimization and the corresponding values of KL divergence, i.e. training accuracy. From this data, we can draw the following conclusions, that hold for both cases under examination:

- Given a circuit depth and a target accuracy, MPS is able to reach the desired KL divergence value with very few iterations, while the QCBM requires many more optimization steps.

- Given a circuit depth, we observe that in some cases the QCBM is able to reach a lower KL divergence than MPS, provided we use a sufficiently high number of iterations. Additional numerics suggests that the QCBM is able to beat the training accuracy of MPS, which instead reaches a plateau that prevents the KL value to decrease further when the number of the optimization steps is increased.

- Given a fixed number of qubits, we compare MPS and QCBM depths in terms of CNOTs. With small n_{layers} or D, we see that MPS is shallower than the QCBM, but as n_{layers} or D are increased there is an inversion point - whose exact location depends on the circuit size - where QCBM becomes shallower than MPS.

We note that some of the recent works on using MPS for describing continuous probability distributions [43] can significantly improve the training cost of MPS or avoid training altogether. This will also affect the comparison with QCBM.

MPS	D	CNOT	1-qb	n_{iters}	KL
$n_{\text{qubits}} = 4$	2	9	19	2	$1.15 \cdot 10^{-4}$
	4	[28, 40]	[42, 80]	2	$3.21 \cdot 10^{-6}$
	6	[61, 100]	[85, 184]	2	$3.21 \cdot 10^{-6}$
	8	[61, 100]	[85, 184]	2	$3.21 \cdot 10^{-6}$
	10	-	-	-	-
$n_{\text{qubits}} = 6$	2	15	31	2	$3.68 \cdot 10^{-2}$
	4	[56, 80]	[84, 160]	2	$2.31 \cdot 10^{-4}$
	6	[183, 300]	[255, 552]	2	$2.93 \cdot 10^{-5}$
	8	[183, 300]	[255, 552]	2	$4.53 \cdot 10^{-5}$
	10	[504, 888]	[682, 1568]	2	$4.53 \cdot 10^{-5}$

TABLE IV: For MPS-equivalent circuit with $n_{qubits} = 4$ (top) and $n_{qubits} = 6$ (bottom), the table displays the circuit details (bond dimension D, number of CNOT gates and number of single qubit gates), the number of optimization steps (i.e. n_{iters}) and the KL divergence values. Since it remains unknown how many CNOT and single qubit gates are needed exactly to implement a general k-qubit unitary operator, we can only give lower and upper bounds (except for the case $D = 2$). Lastly, for $n_{\text{qubits}} = 4$, the upper bound for D is 8: for $n_{qubits} = 4$ and $D > 8$ the formula for the equivalent circuit breaks down. See Appendix B for details about encoding MPS into quantum circuits.

QCBM	n_{layers}	CNOT	1-qb	n_{iters}	KL
$n_{\text{qubits}} = 4$	2	12	38	10	$4.49 \cdot 10^{-1}$
				100	$1.16 \cdot 10^{-1}$
				1500	$1.10 \cdot 10^{-3}$
	4	24	80	10	$6.78 \cdot 10^{-1}$
				100	$1.28 \cdot 10^{-1}$
				4000	$1.82 \cdot 10^{-7}$
	6	36	118	10	$5.86 \cdot 10^{-1}$
				100	$2.95 \cdot 10^{-1}$
				4000	$3.02 \cdot 10^{-10}$
	8	48	156	10	$6.50 \cdot 10^{-1}$
				100	$1.64 \cdot 10^{-1}$
				4000	$8.96 \cdot 10^{-9}$
	10	60	194	10	$5.43 \cdot 10^{-1}$
				100	$1.77 \cdot 10^{-1}$
				5000	$1.08 \cdot 10^{-7}$
$n_{\text{qubits}} = 6$	2	30	87	10	1.088
				1000	$4.43 \cdot 10^{-2}$
				15000	$2.56 \cdot 10^{-2}$
	4	60	180	10	1.204
				1000	$5.81 \cdot 10^{-2}$
				30000	$1.43 \cdot 10^{-3}$
	6	90	267	10	$9.82 \cdot 10^{-1}$
				1000	$6.92 \cdot 10^{-2}$
				50000	$1.03 \cdot 10^{-4}$
	8	120	354	10	$8.81 \cdot 10^{-1}$
				1000	$2.08 \cdot 10^{-1}$
				15000	$2.93 \cdot 10^{-6}$
	10	150	441	10	$8.69 \cdot 10^{-1}$
				1000	$4.48 \cdot 10^{-1}$
				20000	$4.56 \cdot 10^{-6}$

TABLE V: For QCBM with $n_{qubits} = 4$ (top) and $n_{qubits} = 6$ (bottom), the table displays the circuit details (number of layers, number of CNOT gates and number of single qubit gates), the number of optimization steps (i.e. n_{iters}) and the KL divergence values.

IV.2. Controlled rotations

In preparing the quantum state for the CVA problem we are faced with the problem of constructing operators that are all of the following form:

$$R_f : |i\rangle |0\rangle \longmapsto |i\rangle \left(\sqrt{1 - f(x_i)} |0\rangle + \sqrt{f(x_i)} |1\rangle \right), \quad (28)$$

where the function $f : \Omega \longmapsto [0, 1]$ and $x_i \in \Omega$ are discrete points chosen in its domain and lastly, the label $i \in \mathbb{N}$ is an integer indexing discrete points whose binary expansion is $\sum_{k=0}^{k=n+m-1} 2^k i_k$, with $i_k \in [0, 1]$. If f is efficiently computable classically with $\mathcal{O}(\text{poly}(n, m))$ resource, a common strategy for realizing f exactly is by Reversible Logic Synthesis ([32, 33]); the cost of doing so would also be $\mathcal{O}(\text{poly}(m, n))$.The polynomial scaling is efficient in theory but in practice - and especially for near-term quantum computers - much more is to be desired in terms of lowering the circuit cost. For example, for the CVA instance considered in this work (with a total of 7 qubits), RLS requires a total of 89 CNOT gates. For the purpose of a near term alternative to RLS, we introduce an ansatz, *Controlled Rotation Circuit Ansatz* (CRCA) to try to approximate (28) in the following ansatz (Figure

10):

$$\widetilde{R}_f : |i\rangle|0\rangle \longmapsto |i\rangle\left(\sqrt{1-\widetilde{f}(x_i,\vec{\theta})}\,|0\rangle + e^{i\phi(x_i,\vec{\theta})}\sqrt{\widetilde{f}(x_i,\vec{\theta})}\,|1\rangle\right) \quad (29)$$

where $\widetilde{f}$ is some function with the same domain and co-domain as f, and ϕ is some relative phase that depends on x_i and $\vec{\theta}$. This ansatz can be used to approximate the operators R_v, R_q, and R_p (see Equations (12), (13), (14) in Section III). Let $\widetilde{R}_v$, $\widetilde{R}_q$ and $\widetilde{R}_p$ denote approximations to each of the operators in the form described in Equation (29). In the computational basis $\text{span}\{|i\rangle\} \otimes \text{span}\{|0\rangle, |1\rangle\}$ the operators R_f and $\widetilde{R}_f$ can be arranged as block diagonal matrices with each block being an SU(2) rotation $U^{(i)}$ indexed by the label i:

$$R_f = \begin{pmatrix} U^{(1)} & & & \\ & U^{(2)} & & \\ & & \ddots & \\ & & & U^{(2^{n+m})} \end{pmatrix}, \quad (30)$$

$$U^{(i)} = \begin{pmatrix} \sqrt{1-f(x_i)} & -\sqrt{f(x_i)} \\ \sqrt{f(x_i)} & \sqrt{1-f(x_i)} \end{pmatrix}. \quad (31)$$

For $\widetilde{R}_f$ we can similarly consider the block structure in (30) with each block realizing the single-qubit rotation that leads to (29). However, the relative phase factor $e^{i\phi(x_i,\vec{\theta})}$ is immaterial for the purpose of the CVA calculation, since ultimately the quantity desired $\langle\xi|\Pi|\xi\rangle$ (Section III) is independent of the phase factors. We make a simplification and consider an operator $\widetilde{R}'_f$ which is equivalent to $\widetilde{R}_f$ for the purpose of CVA calculation but consists of only real elements:

$$\widetilde{R}'_f = \begin{pmatrix} V^{(1)} & & & \\ & V^{(2)} & & \\ & & \ddots & \\ & & & V^{(2^{n+m})} \end{pmatrix}, \quad (32)$$

$$V^{(i)} = \begin{pmatrix} \sqrt{1-\widetilde{f}(x_i,\vec{\theta})} & -\sqrt{\widetilde{f}(x_i,\vec{\theta})} \\ \sqrt{\widetilde{f}(x_i,\vec{\theta})} & \sqrt{1-\widetilde{f}(x_i,\vec{\theta})} \end{pmatrix}. \quad (33)$$

The error in CRCA training can then be characterized as the norm difference between the unitary operators R_f and $\widetilde{R}'_f$. Based on the block structures in Equations (30) and (32), the norm difference becomes

$$\epsilon_{\text{CRCA},f} = \left\| R_f - \widetilde{R}'_f \right\|_2 = \max_i \|U^{(i)} - V^{(i)}\|_2. \quad (34)$$

Another common way to measure the approximation error of $\widetilde{f}$ with respect to f is by the 1-norm difference:

$$\frac{1}{2^{n+m}} \sum_{i=0}^{2^{n+m}-1} |\widetilde{f}(x_i,\vec{\theta}) - f(x_i)|. \quad (35)$$

In Appendix C we show that $|\widetilde{f}(x_i,\vec{\theta}) - f(x_i)| \leq \|U^{(i)} - V^{(i)}\|_2$, which implies that $\epsilon_{\text{CRCA},f}$ is an upper bound to the 1-norm difference.

Having proposed CRCA as an alternative to RLS, it is instructive to investigate the cost of RLS not only to highlight the value of having CRCA, but also to look into what RLS produces in terms of the number of entangling operations (CNOT gates) for a practical problem. We use the transformation based synthesis method [44] for our implementation of RLS. We assume the existence of three qubit registers, namely:

(i) First register stores the domain values,

(ii) Second register stores the function values,

(iii) Third register contains the ancilla qubit for which the probability of measuring it in the $|1\rangle$ state gives the value of the function.

More concretely, we have the following sequence of quantum operations:

$$U_f : |i\rangle|0\rangle|0\rangle \longmapsto |i\rangle|f(i)\rangle|0\rangle, \quad (36)$$

$$O_f : |i\rangle|f(i)\rangle|0\rangle \longmapsto |i\rangle|f(i)\rangle(a_i|0\rangle + b_i|1\rangle), \quad (37)$$

$$U_f^\dagger : |i\rangle|f(i)\rangle(a_i|0\rangle + b_i|1\rangle) \longmapsto |i\rangle|0\rangle(a_i|0\rangle + b_i|1\rangle), \quad (38)$$

where b_i is such that $|b_i|^2 = f(i)$. The quantum operation U_f is implemented by RLS while O_f is a quantum circuit that contains $\mathcal{O}(n)$ controlled rotations, where n is the number of qubits in the first register. Let C be the number of CNOT gates from RLS, then we need a total of $2C$ CNOT gates to implement the evaluation of the function i.e (36) and its un-computation i.e (38). The number we report for C will contain all the CNOT gates needed to implement functions for the payoff function, the discount factor, and default probabilities. For the step in (37) we require n two qubit gates for both the discount factor and the default probabilities and $2n$ two qubit gates for the payoff function. Thus, the total number of two qubit gates N_{2q} needed to solve a CVA problem instance is

$$N_{2q} = 2C + n + n + 2n = 2(C + 2n). \quad (39)$$

In Table VI we present numerical results for calculating N_{2q} for different numbers of qubits. In comparison, to implement all the three desired functions for the CVA instance with CRCA, we need a total of 6 CNOTs for both the discount factor and the default probabilities and 24 CNOTs (2 layers) for the payoff function, giving a grand total of 36 CNOTs for a qubit size register of 4 qubits. Of course, while we expect to continue to see savings in the number of CNOT gates, the classical optimization problem for CRCA gets harder with the number of qubits.

IV.3. Noiseless quantum CVA value

We now have all the ingredients needed to build the quantum circuit shown in Figure 3 so that we are able

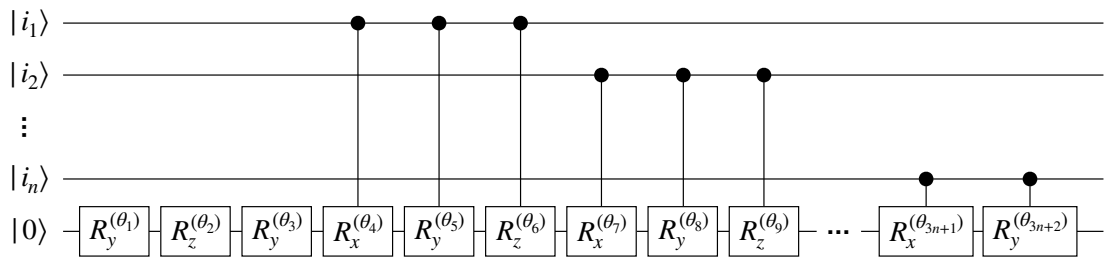


FIG. 10: Controlled Rotation Circuit Ansatz (CRCA), the proposed near term circuit to implement operators of the form in Equation (29).

n	N_{2q}
4	212
6	640
8	1428
10	25226
12	114632
14	483436

TABLE VI: Cost of RLS. n is the number of qubits, while N_{2q} is the number of required two-qubit gates.

Discretization error ϵ_D	$4.376 \cdot 10^{-5}$
Noiseless quantum circuit error ϵ_Q	$7.638 \cdot 10^{-6}$
Noiseless observable error ϵ_Π	$8.836 \cdot 10^{-3}$
State preparation error $\mathrm{KL}(p_G \| p_{tg})$	$4.150 \cdot 10^{-4}$
Payoff function error $\epsilon_{\mathrm{CRCA},v}$	$3.218 \cdot 10^{-3}$
Discount factor error $\epsilon_{\mathrm{CRCA},q}$	$1.545 \cdot 10^{-4}$
Default probability error $\epsilon_{\mathrm{CRCA},p}$	$1.546 \cdot 10^{-4}$

TABLE VII: Error quantities of the quantum circuit for the CVA instance described in Table II.

to get a quantum estimate for the CVA value. Once all the components are trained we can simply run the circuit and calculate the probability of three ancilla being in the state $|111\rangle$. Let $\left|\widetilde{\xi}\right\rangle$ be the actual output state of the quantum circuit (versus the ideal state $|\xi\rangle$ in Equation 18). The quantum CVA value is then:

$$\widetilde{\mathrm{CVA}}_Q = 2^m(1-R)C_pC_qC_v \left\langle\widetilde{\xi}\right| \Pi \left|\widetilde{\xi}\right\rangle \tag{40}$$

$$= 1.987 \cdot 10^{-5} \tag{41}$$

where the values of C_p, C_v, m, C_q are reported in Table II and R is reported in Table I. The above result is calculated from an exact simulation of the quantum circuit. The error in the CVA calculation can be decomposed as the following:

$$\begin{aligned} &|\mathrm{CVA}_{\mathrm{MC}} - \widetilde{\mathrm{CVA}}_Q| \leq \\ &\quad |\mathrm{CVA}_{\mathrm{MC}} - \widetilde{\mathrm{CVA}}(2)| + |\widetilde{\mathrm{CVA}}(2) - \widetilde{\mathrm{CVA}}_Q| \\ &\quad = \epsilon_D + \epsilon_Q \end{aligned} \tag{42}$$

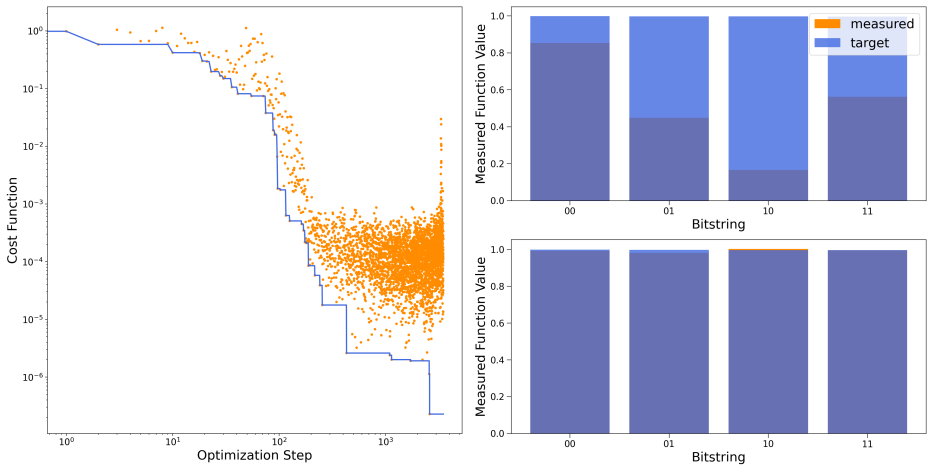
where $\epsilon_D = |\mathrm{CVA}_{\mathrm{MC}} - \widetilde{\mathrm{CVA}}(2)|$ is the discretization error and $\epsilon_Q = |\widetilde{\mathrm{CVA}}(2) - \widetilde{\mathrm{CVA}}_Q|$ is defined as the error due to deviation of the trained quantum circuit from an ideal quantum circuit that prepares $|\xi\rangle$ in Equation (18). With a slight abuse of notation regarding the operators forming the quantum circuit (Figure 3), we consider $|\xi\rangle = R_pR_qR_vG_{\mathcal{P}}|0^{n+m+3}\rangle$ and $|\widetilde{\xi}\rangle = \widetilde{R}_p\widetilde{R}_q\widetilde{R}_v\widetilde{G}_{\mathcal{P}}|0^{n+m+3}\rangle$ where $\widetilde{R}_p$, $\widetilde{R}_q$, $\widetilde{R}_v$, and $\widetilde{G}_{\mathcal{P}}$ denote the operators produced from training. The core component of understanding ϵ_Q is to bound the error in estimating Π, for which we have the following upper bound (derived in Appendix D):

$$\begin{aligned} &\epsilon_\Pi = |\langle\xi|\Pi|\xi\rangle - \langle\widetilde{\xi}|\Pi|\widetilde{\xi}\rangle| \leq \\ &\quad \sqrt{2 \cdot \mathrm{KL}(p_G \| p_{tg})} + 2(\epsilon_{\mathrm{CRCA},v} + \epsilon_{\mathrm{CRCA},q} + \epsilon_{\mathrm{CRCA},p}) \end{aligned} \tag{43}$$

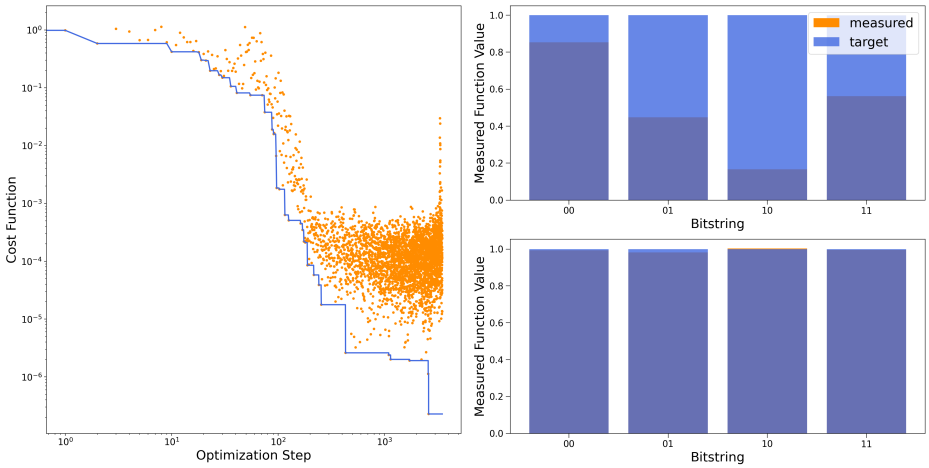
where $\epsilon_{\mathrm{CRCA},f} = \|\widetilde{R}_f - R_f\|_2$. Here we use p_G to denote the output probability distribution (Equation 23 and $p_A = |\Psi_A|^2$ for Ψ_A in Equation 26) from the state preparation operator $G_{\mathcal{P}}$. Although slightly different objective functions are used for QCBM and MPS training, in both cases ($C(\vec{\theta})$ in Equation 24 and $\mathcal{L}$ in Equation 27) they are related to the KL divergence between the generated distribution p_G and the target distribution p_{tg}.

In Table VII we list the error quantities that are useful for gaining insight on the main sources error. Clearly, since the CVA instance contains only $n+m=4$ qubits, the dominant source of error is discretization ϵ_D (computed from Equations 9 and 21), compared with error in building the quantum circuit ϵ_Q (computed from Equations 21 and 41). However, as shown in Figure 2, ϵ_D can be quickly suppressed by increasing the number of qubits n for encoding the asset price. Through Equation (40) we can obtain the error $\epsilon_\Pi = \epsilon_Q/(M(1-R)C_vC_pC_q)$ in estimating the observable Π as listed in Table VII. Based on the upper bound of ϵ_Π, apparently the contribution from state preparation, which is $\sqrt{2 \cdot \mathrm{KL}(p_G \| p_{tg})} = 0.0288$, dominates over the contributions from CRCA training. The value of ϵ_Π is well within the upper bound in (43).

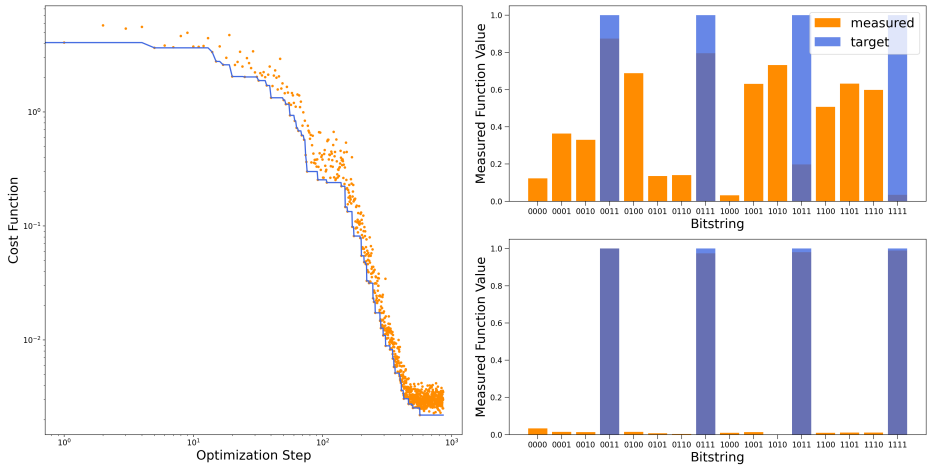
The analysis so far assumes perfect amplitude estimation, namely that we are able to obtain $\langle\widetilde{\xi}|\Pi|\widetilde{\xi}\rangle$ exactly. In reality, with both quantum amplitude estima-



(a) Discount Factor



(b) Default probability



(c) Payoff function

FIG. 11: The CVA circuit requires that we learn functions for **(a)** the discount factor $\tilde{p}(t_i)$, **(b)** the probability of default $\tilde{q}(t_i)$ and **(c)** the payoff function $\tilde{v}(s_j, t_i)$. For each subplot, on the right, the histograms display the functions encoded in the quantum state after the first training step and the one after the final step of training. The plots to the left display the value of the cost function in the training procedure.

tion and classical Monte Carlo algorithms, there is always a statistical error ϵ_S due to the finite amount of computational resource, be it quantum circuit runs or Monte Carlo steps, being used for statistical estimation. In the next section, we discuss using a recently developed Bayesian amplitude estimation technique [8] for performing the amplitude estimation. A central object of this algorithm is the *engineered likelihood function* (ELF) which is used for carrying out Bayesian inference. See Appendix E for a detailed description of this algorithm.

V. CONCRETE RESOURCE ESTIMATION

In this section, we evaluate and compare the runtimes of classical and quantum algorithms for solving the CVA instance specified in Table I. For the classical benchmark, we compute the CVA value based on Equation (1). In particular, to compute the expected exposure $E(t_i)$ at each time t_i, we run a classical Monte Carlo simulation with 10, 100, 1000, 10000 or 100000 paths, respectively. This leads to different runtimes of the algorithm and different errors in the outputs, which are illustrated in Figure 12.

For the quantum amplitude estimation, we apply the ELF-based amplitude estimation algorithm (Appendix E) to the circuit $A = \widetilde{R}_p \widetilde{R}_q \widetilde{R}_v \widetilde{G}_{\mathcal{P}}$ and the projection operator Π given by Equation (19). The ELF-based estimation scheme requires implementing two types of reflection operations:

1. We implement the unitary operator $R_0(y) = \exp(iy \left|0^{n+m+3}\right\rangle \left\langle 0^{n+m+3}\right|)$ with the method in [45], which requires $n+m+2$ ancilla qubits and $8(n+m+2)$ two-qubit gates [46].

2. To implement the operator $U(x) = \exp(ix\Pi)$, we decompose it into 4 two-qubit gates, as implied by Theorem 8 of [42].

As a consequence, the quantum circuit for drawing samples that correspond to an ELF acts on 13 logical qubits and contains $M = 136$ logical two-qubit gates per layer.

We assume that the quantum device has gate error rate 10^{-3} and uses the surface code in [47] for quantum error correction, and we follow the method in [48] to analyze the overhead of this scheme. When the distance of the surface code is d, each logical qubit is mapped to $2d^2$ physical qubits, the fidelity of each non-Clifford gate is $f_{nc} \approx (1-10^{-(d+3)/2})^{100d}$, and the execution time of this gate is $t_{nc} \approx 100d\times$the surface code cycle time. Under a reasonable assumption about the layout of the circuit [48], we have $f_{2Q} \approx f_{nc}$, and $t_{2Q} \approx t_{nc}$. We set the surface code cycle time to $1\mu s$ (which is an optimistic estimate [47]), and vary the code distance from 10 to 26. This leads to different fidelities and execution times of logical two-qubit gates. We also assume the readout fidelity is $\bar{p} = f_{nc}^3$, as the projection operator Π acts on three logical qubits. Under these assumptions about the hardware, we use the following equation to estimate the quantum runtimes needed to achieve the same error tolerance ϵ as in the classical experiments.

$$T_\varepsilon \sim O\left(t_{2Q} M \cdot \frac{e^{-\lambda}}{\bar{p}^2}\left(\frac{\lambda}{\varepsilon^2} + \frac{1}{\sqrt{2}\varepsilon} + \sqrt{\left(\frac{\lambda}{\varepsilon^2}\right)^2 + \left(\frac{2\sqrt{2}}{\varepsilon}\right)^2}\right)\right), \tag{44}$$

where t_{2Q} is the two-qubit gate time, M is the number of two-qubit gates per layer [49], $\lambda = M\ln(1/f_{2Q})$ in which f_{2Q} is the two-qubit gate fidelity, and $\bar{p}$ is the readout fidelity. In the low-noise limit, i.e. $\lambda \ll \epsilon$, Equation (44) recovers the Heisenberg-limit scaling $O(1/\epsilon)$; while in the high-noise limit, i.e. $\lambda \gg \epsilon$, Equation (44) recovers the shot-noise-limit scaling $O(1/\epsilon^2)$. Thus, this model interpolates between the two extreme cases as a function of λ. Such bounds allow us to make concrete statements about the extent of quantum speedup as a function of hardware specifications (e.g. the number of qubits and the two-qubit gate fidelity), and estimate runtimes using realistic parameters for current and future hardware.

The numerical results generated from the runtime model are demonstrated in Figure 12. We observe that for estimating CVA to a relatively large error (e.g. $\geq 0.35\%$), our quantum algorithm runs slower than the classical algorithm. This is mainly because elementary quantum operations take much longer time to execute than their classical counterparts, due to the overhead from quantum error correction. So even though our algorithm contains fewer elementary operations than the classical algorithm, its runtime is still larger than the latter's. Nevertheless, as demonstrated by Figure 13, the runtime of classical algorithm scales almost quadratically in the inverse error in the result, while the runtime of our algorithm scales almost linearly in the same quantity, assuming the gate fidelity is sufficiently high. Therefore, as the desired accuracy of the result becomes higher, the gap between the quantum and classical runtimes will shrink, and eventually the quantum algorithm will surpass the classical one in efficiency. For example, based on the projection in Figure 13, our algorithm will run faster than the classical one for estimating CVA within relative error $\leq 0.0067\%$. We emphasize that this threshold heavily depends on the hardware specifications, and could be dramatically shifted once we have better technology for realizing high-fidelity quantum gates.

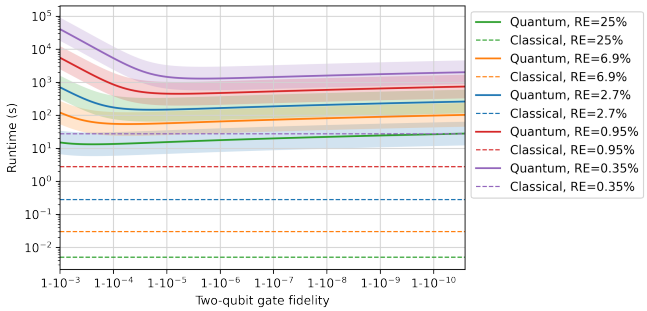


FIG. 12: Classical and quantum runtimes needed to estimate CVA to various accuracies. Here "RE" is short for relative error. Each color represents a target accuracy, and the dashed line and solid curve of that color represent the classical and quantum runtimes needed to reach that accuracy, respectively. We assume that the quantum device uses the surface code in [47] for quantum error correction and follow the method in [48] to analyze the overhead of this scheme. The surface code cycle time is set to $1\mu s$.

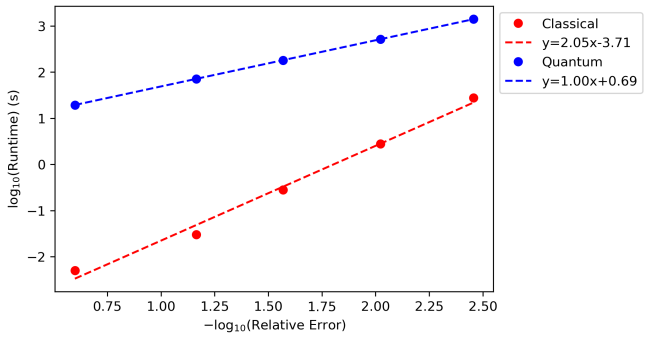


FIG. 13: Trends of classical and quantum runtimes for estimating CVA to higher and higher accuracies. Here we make the same assumption about the quantum device as in Figure 12, setting the code distance to 18. The classical runtime scales almost quadratically in the inverse error in the result, while the quantum runtime scales almost linearly in the same quantity.

Our results are consistent with the ones in [50] which show that it is unlikely to realize quantum advantage on a modest fault-tolerant quantum computer with quantum algorithms giving quadratic speedup over its classical alternatives, due to the large overheads associated with quantum error correction. For such quantum algorithms to run faster than their classical competitors, the input instance must be sufficiently large (as measured by the inverse accuracy of the result in this work), so that the small quantum scaling advantage can compensate for the large overhead factor from error correction. These findings suggest that we should either focus beyond quadratic speedups or dramatically improve the techniques for quantum error correction (or do both) in order to achieve quantum advantage on early generations of fault-tolerant quantum computers.

While our observation of quantum advantage happening at large scales echoes the point in [50], our resource estimation is less pessimistic than the ones in that paper, due to the innovations in our circuit construction. Specifically, [50] has focused on the case where both classical and quantum algorithms make calls to certain primitive circuits to solve the same problem, and the quantum primitive circuit is simply the coherent version of the classical one and is usually obtained via reversible logic synthesis. (Namely, the quantum algorithm has essentially the same spirit as the classical one but leverages amplitude amplification to achieve quadratic speedup.) Due to the overhead from reversible logic synthesis and quantum error correction, the quantum primitive circuit is much slower than its classical counterpart. However, this is not the case in this work, as we have invented alternative, shallower quantum circuits for CVA evaluation that are qualitatively different from the classical one based on Monte Carlo sampling, and consequently, our primitive calls take much less time than the ones considered in [50], which reduces the gap between the runtimes of classical and quantum algorithm at small scales.

VI. DISCUSSION

In this work we have described a general quantum algorithm for computing credit valuation adjustment and performed concrete resource estimation using a specific instance of the CVA problem along with the recently developed engineered likelihood function technique for quantum amplitude estimation. The study has revealed both challenges towards quantum advantage and opportunities for improving the quantum algorithm. Results from Section V show that unless very small statistical error ϵ_S is desired (roughly on the order of 10^{-10}) in estimating $\widetilde{\mathrm{CVA}}_Q$, the classical Monte Carlo implementation is faster than the quantum algorithm. Here we assume that ϵ_S is the dominant source of error, namely that all of the other error quantities in Table VII are significantly smaller than ϵ_S. Fulfilling this assumption requires further innovation on the quantum circuit construction. For instance, in order to suppress ϵ_D we need to increase the number of qubits encoding time and price parameters, which implies that scalable circuit generation training methods for both state preparation and control rotation are needed. The circuit generation methods are not only measured in terms of the scaling of their computational time but also the accuracy of the resulting circuit constructions, since this directly influences the ϵ_Q contribution. We have identified some of the recently developed tools [43, 51, 52] that may help improving circuit generation, and active work is in progress to deploy and test those tools.

From a quantitative finance perspective, it is well-known that an European option pricing problem involving a single asset is analytically solvable, without the need for a Monte Carlo simulation. Nonetheless, our method can be generalized to other settings that do require stochastic simulation, such as multi-asset options, fat-tail distributions for $P(s,t)$, and stochastic intensity

models [53]. As the landscape of risk analysis use cases in quantitative finance unfolds, we expect more innovations to be made in the future that push the frontier of quantum computing closer to practical advantage.

VII. ACKNOWLEDGEMENTS

We thank Brian Dellabetta, Jerome Gonthier, Peter Johnson, Alejandro Perdomo-Ortiz, Max Radin for helpful discussions and support during the project. We thank the team at BBVA for supporting the early parts of this study. All of the numerical experiments in this work were carried out with Orquestra® [54] for workflow and data management.

Appendix A: QCBM Efficiency of Learning Discrete Distributions

In Section IV.1.1, we showed that a Quantum Circuit Born Machine defined on 4 qubits is able to learn the desired target distribution associated to the chosen problem instance. In order to perform the QCBM training, it is essential to collect a certain amount M of measurements outcomes that build up the measured distribution to be compared to the target one. The QCBM training performance changes as the number of shots taken from the quantum computer varies. We performed an initial study to assess the efficiency of the algorithm in dependence on the number of quantum samples M for different circuit sizes and depths. In order to quantify the QCBM efficiency, let's define the following measures:

$$\begin{aligned} \epsilon_\mu &= |\mu_m - \mu_t| \\ \epsilon_{\sigma^2} &= |\sigma_m^2 - \sigma_t^2|, \end{aligned} \tag{A1}$$

where μ and σ^2 are the mean and the variance of the measured (m) and the target (t) distribution. For simplicity, we limited the target distribution to one single log-normal peak. We computed the median of this quantity over 10 equivalent simulations for different value of M and compared the scaling behaviour against the one obtained for a classical Monte Carlo simulation (i.e. rejection sampling).

The relation between QCBM and Monte Carlo scaling depends on the circuit details. Specifically, if one increases the problem size while keeping the depth fixed, the QCBM efficiency decreases, whereas if one increases the depth keeping the number of qubits fixed, the QCBM efficiency increases and eventually outperforms its classical Monte Carlo counterpart. This initial analysis shows that the "aspect ratio" of the quantum circuit matters when it comes to the efficiency of learning a discrete distribution. That is to say: given a n qubits QCBM, it needs to have $f(n)$ layers in order to yield an advantage over Monte Carlo (see Table VIII).

n	2	3	4	5	6
$f(n)$	< 2	< 2	$\gtrsim 4$	≥ 5	≥ 7

TABLE VIII: Relation between number of qubits n and number of layers $f(n)$ needed in the quantum circuit to yield advantage over the classical algorithm.

Appendix B: Quantum Circuits for Preparing Matrix Product States

Our method for encoding matrix product states into quantum circuits is based on the one in [55]. Suppose $|\psi\rangle$ is an MPS given by

$$|\psi\rangle = \langle\psi_F| A^{[1]} A^{[2]} \dots A^{[n]} |\psi_I\rangle \tag{B1}$$

$$= \sum_{i_1,i_2,\dots,i_n \in \{0,1\}} \langle\psi_F| A^{[1]}_{i_1} A^{[2]}_{i_2} \dots A^{[n]}_{i_n} |\psi_I\rangle |i_1 i_2 \dots i_n\rangle , \tag{B2}$$

where $A^{[j]} : \mathbb{C}^D \to \mathbb{C}^D \otimes \mathbb{C}^2$ satisfies $A^{[j]} = A_0^{[j]} \otimes |0\rangle + A_1^{[j]} \otimes |1\rangle$, for $j = 1, 2, \dots, n$, and $|\psi_I\rangle$, $|\psi_F\rangle \in \mathbb{C}^D$ are arbitrary. Without loss of generality, we assume that $D = 2^d$ for some $d \in \mathbb{Z}^+$ (if D is not a power of two, we can embed this MPS into a larger MPS whose bond dimension is $2^{\lceil \log_2(D) \rceil}$). We will find isometries $V^{[1]}$, $V^{[2]}$, $\dots$, $V^{[n]}$ such that

$$|\psi\rangle = V^{[1]} V^{[2]} \dots V^{[n]} |\phi\rangle \tag{B3}$$

for some state $|\phi\rangle \in \mathbb{C}^D$ by a sequence of singular value decompositions (SVDs). Specifically, we start by writing

$$\langle\psi_F| A^{[1]} = V^{[1]} M^{[1]}, \tag{B4}$$

where $V^{[1]}$ is the left unitary matrix in the SVD of the left-hand side, and $M^{[1]}$ is the remaining part of the SVD. Then we construct the other isometries by the following induction:

$$(M^{[k]} \otimes I) A^{[k+1]} = V^{[k+1]} M^{[k+1]}, \tag{B5}$$

where $V^{[k+1]}$ comes from the left unitary matrix in the SVD of the left-hand side, and $M^{[k+1]}$ is the associated remaining part of the SVD. After n applications of SVDs from left to right, we set

$$|\phi\rangle = M^{[n]} |\psi_I\rangle , \tag{B6}$$

obtaining Eq. (B3) as desired. It is easy to show that $V^{[k]}$ has dimension $\min(2D, 2^k) \times \min(D, 2^k)$. This implies that we can embed $V^{[k]}$ into a $2D$-dimensional or 2^k-dimensional unitary $U^{[k]}$, depending on whether $k > d$ or not. (Precisely, we define $U^{[k]}$ as follows. If $k \leq d$, then $U^{[k]} = V^{[k]}$; otherwise, $U^{[k]}$ can be any $2D$-dimensional unitary operator such that $U^{[k]} |0\rangle |\eta\rangle = V^{[k]} |\eta\rangle$ for all $|\eta\rangle \in \mathbb{C}^D$.)

Now we treat $|\phi\rangle$ as a d-qubit state (i.e. we use d qubits to simulate the D-dimensional system). It follows that

$$|\psi\rangle = U^{[1]} U^{[2]} \dots U^{[n]} \left|0^{n-d}\right\rangle |\phi\rangle , \tag{B7}$$

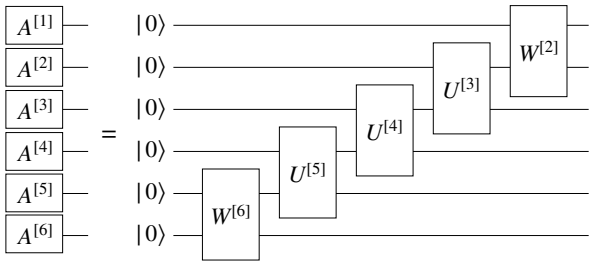


FIG. 14: Quantum circuit for preparing a 6-qubit MPS with bond dimension 2. The two-qubit operators $W^{[2]}$, $U^{[3]}$, $U^{[4]}$, $U^{[5]}$ and $W^{[6]}$ are computed by the procedure in Appendix B.

where $U^{[k]}$ acts on the first k qubits if $k \leq d$, and the $(k-d)$-th to the k-th qubits otherwise. Let Q be a d-qubit unitary operator such that $Q\left|0^d\right\rangle = |\phi\rangle$. Then we get

$$|\psi\rangle = W^{[d+1]}U^{[d+2]}\dots U^{[n-1]}W^{[n]}\left|0^n\right\rangle, \tag{B8}$$

where $W^{[d+1]} := U^{[1]}U^{[2]}\dots U^{[d+1]}$ acts on the first $d+1$ qubits, and $W^{[n]} := U^{[n]}(I \otimes Q)$ acts on the $(n-d)$-th to the n-th qubits. Eq. (B8) leads to a quantum circuit for preparing the state $|\psi\rangle$. Figure 14 demonstrates this circuit for the case $n = 6$ and $D = 2$.

The unitary operators $W^{[d+1]}$, $U^{[d+2]}$, ..., $U^{[n-1]}$ and $W^{[n]}$ can be decomposed into one- and two-qubit gates by using the method in [42]. Since each of these operators acts on $d+1$ qubits, it can be implemented by a circuit containing c_{d+1} CNOT gates, where $[4^{d+1} - 3(d+1) - 1]/4 \leq c_{d+1} \leq (23/48) \times 4^{d+1} - (3/2) \times 2^{d+1} + 4/3$. This implies that the number of CNOT gates in the final circuit is $\Theta(n4^d) = \Theta(nD^2)$. Table IX gives estimated CNOT counts for various n's and D's [56].

	D=2	D=4	D=8	D=16
n=4	9	[28, 40]	–	–
n=8	21	[84, 120]	[305, 500]	[1008, 1776]
n=16	45	[196, 280]	[793, 1300]	[3024, 5328]
n=32	93	[420, 600]	[1769, 2900]	[7056, 12432]

TABLE IX: Estimated numbers of CNOT gates in the quantum circuits for MPS of various sizes and bond dimensions. Here n is the number of qubits, and D is the bond dimension. For each pair (n, D), we give an upper bound and a lower bound on the corresponding CNOT count. Without loss of generality, we assume that $D \leq 2^{\lfloor n/2 \rfloor}$, since any n-qubit MPS can be transformed into an equivalent form with bond dimension at most $2^{\lfloor n/2 \rfloor}$. So there are no entries for $(n = 4, D = 8)$ or $(n = 4, D = 16)$.

Appendix C: Comparing two SU(2) rotations for CRCA

We are concerned with relating the cost function of the form in Equation (35) to the norm difference $\|U - V\|_2$ between SU(2) rotations U and V such that

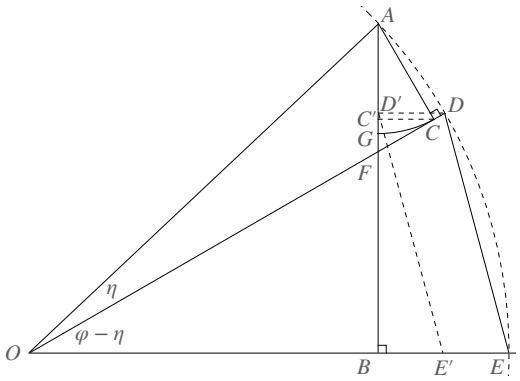


FIG. 15: Geometric proof of inequality (C3).

$$U = \begin{pmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{pmatrix}, \quad V = \begin{pmatrix} \cos\eta & -\sin\eta \\ \sin\eta & \cos\eta \end{pmatrix}. \tag{C1}$$

It is clear that

$$\begin{aligned} \|U - V\|_2 &= \|I - VU^\dagger\|_2 \\ &= \|I - \exp\left(-i(\varphi - \eta)\sigma_y\right)\|_2 \\ &= 2\left|\sin\left(\frac{\varphi - \eta}{2}\right)\right|. \end{aligned} \tag{C2}$$

A quantity that is relevant to the training error of CRCA, as described in Equation (35), is the difference $|\sin\varphi - \sin\eta|$. This quantity is equal to $|\tilde{f}(x_j, \vec{\theta}) - f(x_j)|$ when U is the SU(2) rotation in the block of the CRCA unitary indexed by j and V is the ideal rotation in the same block with all of its matrix elements being real numbers. We show that

$$|\sin\varphi - \sin\eta| \leq 2\left|\sin\left(\frac{\varphi - \eta}{2}\right)\right|, 0 \leq \eta < \varphi \leq \pi/2 \tag{C3}$$

using a geometric illustration (Figure 15). Assume $0 \leq \eta \leq \varphi < \pi/2$. From point O we make three rays $\overrightarrow{OA}$, $\overrightarrow{OD}$ and $\overrightarrow{OE}$ such that $\angle AOD = \eta$ and $\angle DOE = \varphi - \eta$. Points A, D, and E form an arc $\widehat{ADE}$ on a unit circle. Therefore $|\overrightarrow{DE}| = 2|\sin\left(\frac{\varphi-\eta}{2}\right)|$. From A we make two lines $\overrightarrow{AC}$ and $\overrightarrow{AB}$ that are perpendicular to $\overrightarrow{OD}$ and $\overrightarrow{OE}$ respectively. Therefore $|\overrightarrow{AC}| = \sin\eta$ and $|\overrightarrow{AB}| = \sin\varphi$. Using A as the center and $\overrightarrow{AC}$ as the radius, we make an arc $\widehat{CG}$ which intercepts $\overrightarrow{AB}$ at G. Hence

$$|\overrightarrow{BG}| = |\sin\varphi - \sin\eta|. \tag{C4}$$

From points C and D we make perpendicular lines to $|\overrightarrow{AB}|$ that intercepts $\overrightarrow{AB}$ at C' and D' respectively. Since $\eta > 0$, C is between D and F. Since $\overrightarrow{CC'}$ is in parallel to $\overrightarrow{DD'}$, C' is between D' and F. Since $\angle C'CF = \angle DOE = \varphi - \eta$ and $\overrightarrow{CF}$ is a tangent of $\widehat{CG}$, G is between F and

C', which is itself between F and D'. Therefore we have that G is between F and D', namely

$$|\overrightarrow{BG}| < |\overrightarrow{BD'}|. \tag{C5}$$

From D' we make a parallel line to DE which intersects $\overrightarrow{AE}$ at E'. Then

$$|\overrightarrow{BD'}| < |\overrightarrow{E'D'}| = |\overrightarrow{ED}| = 2\left|\sin\left(\frac{\varphi-\eta}{2}\right)\right|. \tag{C6}$$

Combining (C4), (C5) and (C6) yields the conclusion in (C3).

Appendix D: Derivation of the CVA observable error bound

Here we derive the upper bound in (43). We start from state preparation, where we generate $p_G(x_i) = |\langle x_i|G_{\mathcal{P}}|0^{n+m}\rangle|^2$ to approximate a target distribution $p_{tg}(x_i)$ for $x_i \in \Omega = \{0,1\}^{n+m}$, $i = 1, \cdots, 2^{n+m}$. We can then rewrite the error term as

$$\left|\langle\xi|\Pi|\xi\rangle - \langle\widetilde{\xi}|\Pi|\widetilde{\xi}\rangle\right| = \left|\sum_i p_{tg}(x_i)\langle x_i|R_v^\dagger R_q^\dagger R_p^\dagger \Pi R_p R_q R_v|x_i\rangle - \sum_i p_G(x_i)\langle x_i|\widetilde{R}_v^\dagger \widetilde{R}_q^\dagger \widetilde{R}_p^\dagger \Pi \widetilde{R}_p \widetilde{R}_q \widetilde{R}_v|x_i\rangle\right| \tag{D1}$$

$$= \Bigg|\sum_i p_{tg}(x_i)\langle x_i|R_v^\dagger R_q^\dagger R_p^\dagger \Pi R_p R_q R_v|x_i\rangle - \sum_i p_G(x_i)\langle x_i|R_v^\dagger R_q^\dagger R_p^\dagger \Pi R_p R_q R_v|x_i\rangle$$

$$+ \sum_i p_G(x_i)\langle x_i|R_v^\dagger R_q^\dagger R_p^\dagger \Pi R_p R_q R_v|x_i\rangle - \sum_i p_G(x_i)\langle x_i|\widetilde{R}_v^\dagger \widetilde{R}_q^\dagger \widetilde{R}_p^\dagger \Pi \widetilde{R}_p \widetilde{R}_q \widetilde{R}_v|x_i\rangle\Bigg| \tag{D2}$$

$$\leq \sum_i |p_G(x_i) - p_{tg}(x_i)| + \sum_i p_G(x_i)\left|\langle x_i|R_v^\dagger R_q^\dagger R_p^\dagger \Pi R_p R_q R_v|x_i\rangle - \langle x_i|\widetilde{R}_v^\dagger \widetilde{R}_q^\dagger \widetilde{R}_p^\dagger \Pi \widetilde{R}_p \widetilde{R}_q \widetilde{R}_v|x_i\rangle\right| \tag{D3}$$

The first term in (D3) can be bounded from above by $\sqrt{2\cdot \mathrm{KL}(p_G||p_{tg})}$ due to Pinsker's inequality. The second term can be bounded from above by

$$\sum_i p_G(x_i)\Big|\langle x_i|R_v^\dagger R_q^\dagger R_p^\dagger \Pi R_p R_q R_v|x_i\rangle - \langle x_i|\widetilde{R}_v^\dagger \widetilde{R}_q^\dagger \widetilde{R}_p^\dagger \Pi R_p R_q R_v|x_i\rangle + \langle x_i|\widetilde{R}_v^\dagger \widetilde{R}_q^\dagger \widetilde{R}_p^\dagger \Pi R_p R_q R_v|x_i\rangle - \langle x_i|\widetilde{R}_v^\dagger \widetilde{R}_q^\dagger \widetilde{R}_p^\dagger \Pi \widetilde{R}_p \widetilde{R}_q \widetilde{R}_v|x_i\rangle\Big| \tag{D4}$$

$$\leq \sum_i p_G(x_i)\left(\left|\langle x_i|\left(R_v^\dagger R_q^\dagger R_p^\dagger - \widetilde{R}_v^\dagger \widetilde{R}_q^\dagger \widetilde{R}_p^\dagger\right)\Pi R_p R_q R_v|x_i\rangle\right| + \left|\langle x_i|\widetilde{R}_v^\dagger \widetilde{R}_q^\dagger \widetilde{R}_p^\dagger \Pi\left(R_p R_q R_v - \widetilde{R}_p \widetilde{R}_q \widetilde{R}_v\right)|x_i\rangle\right|\right) \tag{D5}$$

$$\leq 2\left\|\widetilde{R}_p \widetilde{R}_q \widetilde{R}_v - R_p R_q R_v\right\|_2 \tag{D6}$$

where going from (D5) to (D6) we apply the following argument:

$$\left|\langle 0|V^\dagger \underbrace{|111\rangle\langle 111|}_{\Pi}(V-U)|0\rangle\right| = \left|\langle 0|V^\dagger \underbrace{\sigma_x^{\otimes 3}|0\rangle}_{=|111\rangle}\right| \cdot \left|\langle 0|\sigma_x^{\otimes 3}(V-U)|0\rangle\right| \tag{D7}$$

$$\leq \|V^\dagger \sigma_x^{\otimes 3}\|_2 \cdot \|\sigma_x^{\otimes 3}(V-U)\|_2$$

$$= \|V-U\|_2.$$

When applying the argument for each term in the sum of (D5) we let U and V be unitary operators such that $U|0\rangle = \widetilde{R}_p\widetilde{R}_q\widetilde{R}_v|x_i\rangle$ and $V|0\rangle = R_pR_qR_v|x_i\rangle$. The operator $\sigma_x^{\otimes 3}$ acts on the three ancilla qubits that the projector Π acts non-trivially on (Equation 19).

The term in (D6) can be further bounded from above by

$$\left\|\widetilde{R}_p\widetilde{R}_q\widetilde{R}_v - R_pR_qR_v\right\|_2 = \left\|\widetilde{R}_p\widetilde{R}_q\widetilde{R}_v - \widetilde{R}_p\widetilde{R}_qR_v + \widetilde{R}_p\widetilde{R}_qR_v - R_pR_qR_v\right\|_2$$

$$= \left\|\widetilde{R}_p\widetilde{R}_q\left(\widetilde{R}_v - R_v\right) + \left(\widetilde{R}_p\widetilde{R}_q - R_pR_q\right)R_v\right\|_2$$

$$\leq \left\|\widetilde{R}_v - R_v\right\|_2 + \left\|\widetilde{R}_p\widetilde{R}_q - R_pR_q\right\|_2 \tag{D8}$$

$$\leq \cdots$$

$$\leq \left\|\widetilde{R}_v - R_v\right\|_2 + \left\|\widetilde{R}_p - R_p\right\|_2 + \left\|\widetilde{R}_q - R_q\right\|_2 \tag{D9}$$

where in going from (D8) to (D9) we essentially repeat the same argument that leads to (D8). Combining (D9) and (D3) yields (43).

Appendix E: Quantum amplitude estimation using engineered likelihood function (ELF)

Here we describe the quantum algorithm in [8] for robust amplitude estimation. Suppose we want to estimate the expectation value

$$\eta = \cos(\theta) = \langle A|\,O\,|A\rangle, \tag{E1}$$

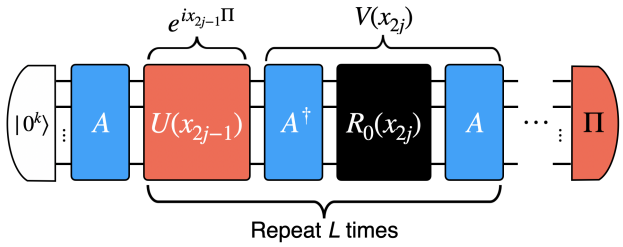


FIG. 16: Quantum circuit for generating samples that correspond to an engineered likelihood function. Here $R_0(y) = \exp(iy\,|0^k\rangle\langle 0^k|)$ in which $y \in \mathbb{R}$ is arbitrary.

where $|A\rangle = A\,|0^k\rangle$ in which A is a k-qubit unitary operator, $O = 2\Pi - I$ in which Π is a projection operator, and $\theta = \arccos(\eta)$ is introduced to facilitate Bayesian inference later on. For the CVA problem, $A = \widetilde{R}_p\widetilde{R}_q\widetilde{R}_v\widetilde{G}_{\mathcal{P}}$ is the quantum circuit for preparing the state $|A\rangle = \left|\widetilde{\xi}\right\rangle$, and Π is given by Equation (19). Then $\langle A|\,O\,|A\rangle = 2\widetilde{\mathrm{CVA}}_Q/C - 1$, where $C = M(1-R)C_vC_pC_q$ by Equation (7). So one can infer $\widetilde{\mathrm{CVA}}_Q$ from the estimate of $\langle A|\,O\,|A\rangle$.

We use the quantum circuit in Figure 16 to generate the *engineered likelihood function* (ELF), which is the probability distribution of a binary outcome $d \in \{0,1\}$ given the unknown quantity θ to be estimated. The circuit consists of a sequence of unitary operations of forms $U(x) = \exp(ix\Pi)$ and $V(y) = A\exp(iy\,|0^k\rangle\langle 0^k|)A^\dagger$ in which $x, y \in \mathbb{R}$ are tunable parameters. Specifically, after preparing the ansatz state $|A\rangle = A\,|0^k\rangle$, we apply $2L$ unitary operations $U(x_1)$, $V(x_2)$, $\ldots$, $U(x_{2L-1})$, $V(x_{2L})$ to it, varying the rotation angle x_j in each operation. For convenience, we call $V(x_{2j})U(x_{2j-1})$ the j-th layer of the circuit, for $j = 1, 2, \ldots, L$. The output state of this circuit is

$$Q(\vec{x})\,|A\rangle = V(x_{2L})U(x_{2L-1})\ldots V(x_2)U(x_1)\,|A\rangle\,, \quad \text{(E2)}$$

where $\vec{x} = (x_1, x_2, \ldots, x_{2L-1}, x_{2L}) \in \mathbb{R}^{2L}$ contains the tunable parameters. Finally, we perform the projective measurement $\{\Pi, I-\Pi\}$ on this state, receiving outcome $d \in \{0,1\}$ with probability

$$\mathbb{P}(d|\vec{x}) = \frac{1 + (-1)^d\,\langle A|\,Q^\dagger(\vec{x})OQ(\vec{x})\,|A\rangle}{2}. \quad \text{(E3)}$$

This Bernoulli distribution depends on θ implicitly.

In practice, the quantum circuit for generating the ELF is inevitably noisy, and the bias of the Bernoulli distribution in Eq. (E3) will be re-scaled by a factor of the fidelity of the circuit. Namely, if the fidelity of the circuit is $f \in [0,1]$, then the probability of obtaining outcome $d \in \{0,1\}$ becomes

$$\mathbb{P}(d|f,\vec{x}) = \frac{1 + (-1)^d f\,\langle A|\,Q^\dagger(\vec{x})OQ(\vec{x})\,|A\rangle}{2}, \quad \text{(E4)}$$

which still depends on θ implicitly.

We use a Gaussian distribution to represent our knowledge of θ and keep updating this distribution until it is sufficiently concentrated around a single value. Specifically, we begin with an initial distribution of η and convert it to the initial distribution of $\theta = \arccos(\eta)$. Then we repeat the following procedure until convergence is reached. At each round, we first compute the circuit parameters $\vec{x} \in \mathbb{R}^{2L}$ that maximize the information gain from the measurement outcome d (in certain sense). Then we run the quantum circuit in Figure 16 with the optimized parameters $\vec{x}$ and receive a measurement outcome $d \in \{0,1\}$. After that, we update the distribution of θ by using Bayes' rule. Once this loop is finished, we convert the final distribution of θ to the final distribution of $\eta = \cos(\theta)$, and set the mean of this distribution as the estimate of η.

We have discovered that the efficiency of this algorithm is determined by the Fisher information of the engineered likelihood function at each round, and proposed efficient heuristic algorithms for finding the parameters $\vec{x} \in \mathbb{R}^{2L}$ that maximize this quantity. We have also found that the engineered likelihood function resembles a sinusoidal function in the critical region, and this fact allows us to perform Bayesian update efficiently without resorting to numerical integration. See [8] for more details.

[1] Basel Committee on Banking Supervision. Targeted revisions to the credit valuation adjustment risk framework. *Bank for International Settlements*, July 2020. https://www.bis.org/bcbs/publ/d507.pdf.

[2] Alvin Lee. The triple convergence of credit valuation adjustment (CVA). 2015. URL: https://www.fixglobal.com/home/the-triple-convergence-of-credit-valuation-adjustment-cva/.

[3] Peter E. Kloeden and Eckhard Platen. *Numerical Solution of Stochastic Differential Equations*. Springer Berlin Heidelberg, 1992.

[4] Michael B. Giles. Multilevel monte carlo methods. *Acta Numerica*, 24:259–328, April 2015.

[5] Gilles Brassard, Peter Hoyer, Michele Mosca, and Alain Tapp. Quantum amplitude amplification and estimation. 2000.

[6] Daniel S. Abrams and Colin P. Williams. Fast quantum algorithms for numerical integrals and stochastic processes, 1999. arXiv:quant-ph/9908083.

[7] Daochen Wang, Oscar Higgott, and Stephen Brierley. Accelerated variational quantum eigensolver. *Physical Review Letters*, 122(14), April 2019.

[8] Guoming Wang, Dax Enshan Koh, Peter D. Johnson, and Yudong Cao. Minimizing estimation runtime on noisy quantum computers. *PRX Quantum*, 2021. In press. arXiv:2006.09350.

[9] Yohichi Suzuki, Shumpei Uno, Rudy Raymond, Tomoki Tanaka, Tamiya Onodera, and Naoki Yamamoto. Amplitude estimation without phase estimation. *Quantum Information Processing*, 19(2), January 2020.

[10] Tudor Giurgica-Tiron, Iordanis Kerenidis, Farrokh Labib, Anupam Prakash, and William Zeng. Low depth algorithms for quantum amplitude estimation, 2020. arXiv:2012.03348.

[11] Dax Enshan Koh, Guoming Wang, Peter D. Johnson, and Yudong Cao. A framework for engineering quantum likelihood functions for expectation estimation, 2020. arXiv:2006.09349.

[12] Lov Grover and Terry Rudolph. Creating superpositions that correspond to efficiently integrable probability distributions, 2002.

[13] Dorit Aharonov and Amnon Ta-Shma. Adiabatic quantum state generation and statistical zero knowledge. In *Proceedings of the Thirty-Fifth Annual ACM Symposium on Theory of Computing*, STOC '03, page 20–29, New York, NY, USA, 2003. Association for Computing Machinery.

[14] Alexei Kitaev and William A. Webb. Wavefunction preparation and resampling using a quantum computer, 2008. arXiv:0801.0342.

[15] Maris Ozols, Martin Roetteler, and Jérémie Roland. Quantum rejection sampling. *ACM Trans. Comput. Theory*, 5(3), August 2013.

[16] Ashwin Nayak and Felix Wu. The quantum query complexity of approximating the median and related statistics. 1998. arXiv:quant-ph/9804066.

[17] Daniel S. Abrams and Colin P. Williams. Fast quantum algorithms for numerical integrals and stochastic processes. 1999. arXiv:quant-ph/9908083.

[18] Erich Novak. Quantum complexity of integration. *Journal of Complexity*, 17(1):2–16, March 2001.

[19] Gilles Brassard, Peter HØyer, and Alain Tapp. Quantum counting. In *Automata, Languages and Programming*, pages 820–831. Springer Berlin Heidelberg, 1998.

[20] Patrick Rebentrost, Brajesh Gupt, and Thomas R. Bromley. Quantum computational finance: Monte carlo pricing of financial derivatives. *Phys. Rev. A*, 98:022321, Aug 2018.

[21] Nikitas Stamatopoulos, Daniel J. Egger, Yue Sun, Christa Zoufal, Raban Iten, Ning Shen, and Stefan Woerner. Option Pricing using Quantum Computers. *Quantum*, 4:291, July 2020.

[22] Stefan Woerner and Daniel J. Egger. Quantum risk analysis. *npj Quantum Information*, 5(1), February 2019.

[23] Ashley Montanaro. Quantum speedup of monte carlo methods. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 471(2181):20150301, September 2015.

[24] 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, 2020. arXiv:2012.06283.

[25] Marco Marchioro. Pricing simple credit derivatives. 2009. http://janroman.dhis.org/finance/Kreditrisk/intro-credit-derivatives.pdf.

[26] https://www.quantlib.org/.

[27] Pawel Wocjan, Chen-Fu Chiang, Daniel Nagaj, and Anura Abeyesinghe. Quantum algorithm for approximating partition functions. *Physical Review A*, 80(2), August 2009.

[28] Vlatko Vedral, Adriano Barenco, and Artur Ekert. Quantum networks for elementary arithmetic operations. *Physical Review A*, 54(1):147–153, July 1996.

[29] David Beckman, Amalavoyal N. Chari, Srikrishna Devabhaktuni, and John Preskill. Efficient networks for quantum factoring. *Physical Review A*, 54(2):1034–1063, August 1996.

[30] Thomas G. Draper, Samuel A. Kutin, Eric M. Rains, and Krysta M. Svore. A logarithmic-depth quantum carry-lookahead adder. *Quantum Info. Comput.*, 6(4):351–369, July 2006.

[31] Steven A. Cuccaro, Thomas G. Draper, Samuel A. Kutin, and David Petrie Moulton. A new quantum ripple-carry addition circuit, 2004.

[32] D. Maslov D. M. Miller and G. W. Dueck. A transformation based algorithm for reversible logic synthesis. In *Proceedings 2003. Design Automation Conference*, pages 318–323. IEEE, aug 2003.

[33] Miller D.M. Soeken M., Dueck G.W. Reversible computation. In Lanese I. Devitt S., editor, *Reversible Computation*, volume 9720 of *Lecture Notes in Computer Science,*. Springer, 2016.

[34] Nathan Wiebe and Martin Roetteler. Quantum arithmetic and numerical analysis using repeat-until-success circuits. *Quantum Info. Comput.*, 16(1–2):134–178, January 2016.

[35] With RLS, consider the state preparation process $|i\rangle|0\rangle \rightarrow |i\rangle|p(i)\rangle \rightarrow |i\rangle|p(i)\rangle(\sqrt{1-p(i)}|0\rangle + \sqrt{p(i)}|1\rangle)$ and post selection on the state $|1\rangle$ for the last qubit. With proper uncomputation, one can then generate the state $\Sigma_i\sqrt{p(i)}|i\rangle$ which is relevant in the context of this paper.

[36] Daiwei Zhu, Norbert M Linke, Marcello Benedetti, Kevin A Landsman, Nhung H Nguyen, C Huerta Alderete, Alejandro Perdomo-Ortiz, Nathan Korda, A Garfoot, Charles Brecque, et al. Training of quantum circuits on a hybrid quantum computer. *Science advances*, 5(10):eaaw9918, 2019.

[37] Lei Wang Han, Wang and Zhang. Unsupervised generative modeling using matrix product states. *arXiv preprint arXiv:1709.01662*, 2018.

[38] Pierre-Luc Dallaire-Demers and Nathan Killoran. Quantum generative adversarial networks. *Physical Review A*, 98(1), July 2018.

[39] Christa Zoufal, Aurélien Lucchi, and Stefan Woerner. Quantum generative adversarial networks for learning and loading random distributions. *npj Quantum Information*, 5(1), November 2019.

[40] Jonathan Romero and Alán Aspuru-Guzik. Variational quantum generators: Generative adversarial quantum machine learning for continuous distributions. *Advanced Quantum Technologies*, 4(1):2000003, December 2020.

[41] Nikolaus Hansen. The cma evolution strategy: a comparing review. In *Towards a new evolutionary computation*, pages 75–102. Springer, 2006.

[42] V. V. Shende, S. S. Bullock, and I. L. Markov. Synthesis of quantum-logic circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25(6):1000–1010, 2006.

[43] Adam Holmes and A. Y. Matsuura. Efficient quantum circuits for accurate state preparation of smooth, differentiable functions, 2020.

[44] Mathias Soeken, Stefan Frehse, Robert Wille, and Rolf Drechsler. Revkit: A toolkit for reversible circuit design. volume 18, 07 2010.

[45] Adam Holmes, Sonika Johri, Gian Giacomo Guerreschi, James S Clarke, and A Y Matsuura. Impact of qubit connectivity on quantum algorithm performance. *Quantum Science and Technology*, 5(2):025009, mar 2020.

[46] The idea of implementing $R_0(y) = \exp(iy\,|0^n+m+2\rangle\langle 0^n+m+2|)$ is as follows. We first compute the logical AND of the bitwise NOT of the input by using $n+m+2$ ancilla qubits and $n+m+2$ Toffoli gates which are organized in a binary-tree fashion. Then we perform a R_z rotation on the ancilla qubit that encodes the result. Finally, we undo the logical AND and bitwise NOT operations, restoring the ancilla qubits to their initial states. Each Toffoli gate can be decomposed into 4 two-qubits gates, as implied by Theorem 8 of [42].

[47] D. S. Wang, A. G. Fowler, A. M. Stephens, and L. C. L. Hollenberg. Threshold error rates for the toric and planar codes. *Quantum Info. Comput.*, 10(5):456–469, May 2010.

[48] William J Huggins, Sam McArdle, Thomas E O'Brien, Joonho Lee, Nicholas C Rubin, Sergio Boixo, K Birgitta Whaley, Ryan Babbush, and Jarrod R McClean. Virtual distillation for quantum error mitigation. *arXiv preprint arXiv:2011.07064*, 2020.

[49] [8] has assumed that the two-qubit gates are arranged in a bricklayer fashion and hence $M \approx nD/2$, where n is the number of qubits and D is the two-qubit gate depth per layer. This assumption does not hold here, as our ansatz circuit is highly sequential.

[50] Ryan Babbush, Jarrod McClean, Craig Gidney, Sergio Boixo, and Hartmut Neven. Focus beyond quadratic speedups for error-corrected quantum advantage. *arXiv preprint arXiv:2011.04149*, 2020.

[51] Frederic Sauvage, Sukin Sim, Alexander A. Kunitsa, William A. Simon, Marta Mauri, and Alejandro Perdomo-Ortiz. Flip: A flexible initializer for arbitrarily-sized parametrized quantum circuits, 2021. arXiv:2103.08572.

[52] Sukin Sim, Jonathan Romero, Jerome F. Gonthier, and Alexander A. Kunitsa. Adaptive pruning-based optimization of parameterized quantum circuits, 2020.

[53] Samim Ghamami and Lisa R. Goldberg. Stochastic intensity models of wrong way risk: Wrong way cva need not exceed independent cva. *Finance and Economics Discussion Series (FEDS)*, 2014. https://www.federalreserve.gov/econres/feds/stochastic-intensity-models-of-wrong-way-risk-wrong-way-cva-need-not-exceed-independent-cva.htm.

[54] https://www.orquestra.io/.

[55] C. Schön, E. Solano, F. Verstraete, J. I. Cirac, and M. M. Wolf. Sequential generation of entangled multiqubit states. *Phys. Rev. Lett.*, 95:110503, Sep 2005.

[56] Here we also give estimated numbers of single-qubit gates in the circuit for preparing a n-qubit MPS with bond dimension $D = 2^d$. If arbitrary single-qubit gates can be used, then the number of single-qubit gates is $6n-5$ if $d = 1$, or between $(n-d)(4^{d+1}-1)/3$ and $(n-d)(13\times 4^{d-1}-3\times 2^d)$ if $d \geq 2$. If only $R_x(\theta)$, $R_y(\theta)$ and $R_z(\theta)$ gates can be used, where $\theta \in \mathbb{R}$ is arbitrary, then the number of single-qubit gates is $12n-9$ if $d = 1$, or between $(n-d)(4^{d+1}-1)$ and $(n-d)(21\times 4^{d-1} - 3\times 2^d))$ if $d \geq 2$.