Quantum Algorithms for Derivative Pricing with Efficient Classical-Quantum Transformations

Kenji Kubo

SEPTEMBER 2022

Quantum Algorithms for Derivative Pricing with Efficient Classical-Quantum Transformations

A dissertation submitted to
THE GRADUATE SCHOOL OF ENGINEERING SCIENCE
OSAKA UNIVERSITY
in partial fulfillment of the requirements for the degree of
DOCTOR OF PHILOSOPHY IN SCIENCE
BY

KENJI KUBO

SEPTEMBER 2022

Abstract

Derivatives are financial instruments whose payments are determined by reference to the prices of more fundamental assets, such as stocks, bonds, or currencies. Since derivatives are financial instruments, their prices need to be calculated, but in general, a large number of computational resources are needed to calculate the prices of derivatives. Therefore, there have been efforts to reduce the amount of computation needed to calculate the prices of derivatives by using a quantum computer. We focus on two approaches to derivatives price calculation using quantum computers. One approach is to use quantum amplitude estimation to speed up Monte-Carlo methods since derivative prices can be calculated as the expected value of the payoff function. The other approach is to use a quantum ordinary differential equation solver to solve the ordinary differential equation obtained by discretizing the partial differential equation that the derivative price obeys. Even when using these quantum algorithms, the transformation between classical and quantum information is often the bottleneck in naive algorithms. In fact, when using the quantum amplitude estimation, embedding the probability distribution into the quantum state can become a bottleneck. When using a quantum ordinary differential equation solver, in addition to embedding the probability distributions and initial values into the quantum state, the extraction of the present price of the derivative from the resulting quantum state also becomes a bottleneck.

In this thesis, we propose quantum algorithms that overcome these bottlenecks in derivative pricing by achieving an efficient classical-quantum transformation. As a method of transformation from classical to quantum information, we propose a simulation of the time evolution of probability distributions using variational quantum simulation. From the stochastic differential equation, we derive a linear ordinary differential equation that the probability distribution obeys and show the way to construct a quantum circuit to simulate the obtained ordinary differential equation by using variational quantum simulation. This allows us to obtain unitary operators for embedding the probability distribution of a Markov process into a quantum state. Compared to other quantum state generation models, this method has the advantage of obtaining the embedding operators at multiple time points in a single simulation. As a method for transformation from quantum

information to classical information, we propose a quantum algorithm based on martingales. A martingale is a property in which the present price of the derivative can be calculated as the expected value of the derivative price at any future point in time. Using a quantum ordinary differential equation solver, we solve the partial differential equation that the derivative price follows to some future point in time instead of solving it to the present. In addition, we prepare quantum states in which the probability distributions at that time is embedded. By calculating the inner product of these quantum states using quantum amplitude estimation, the present price of the derivative can be calculated. Instead of retrieving the present price of the derivative embedded in the amplitude of one basis of the quantum state, the expected value can be calculated using all the amplitudes of the quantum state in this way. Therefore, the bottleneck in the transformation from quantum information to classical information is eliminated, and a significant speed-up can be achieved compared to the classical algorithm. We also propose a method to run this algorithm on a noisy intermediate-scale quantum computer using variational quantum algorithms. The results of this thesis will be an important breakthrough for quantum algorithms in derivatives pricing.

Contents

Chapter 1

Introduction

1.1 Derivative pricing

Derivatives are financial products "derived" from more fundamental assets called the underlying assets, such as stocks, bonds, currency, etc., and have payoffs that depend on the prices of the underlying assets [1, 2]. Because of the uncertainties in the prices of the underlying assets, there are risks in holding such assets. Derivatives are designed to control such risks. Since derivatives are financial instruments, we need to evaluate their prices correctly. To this end, stochastic analysis is used to model the uncertainty of the prices of the assets and the derivatives. This allows us to handle the values and risks as statistics, which can then be quantitatively evaluated. The theoretical price of a derivative in the simplest model can be obtained analytically, but it is necessary to perform numerical calculations in general cases. Financial risks have become increasingly complex and diverse, and a variety of derivatives have been developed to hedge these risks. Accordingly, the calculation of derivative prices has become more complex. In fact, financial institutions use supercomputers to perform these calculations. Moreover, asset prices are constantly varying and then, if real-time recalculation is possible, a higher level of risk management will be achieved. Not only in academia but in practice, derivatives pricing is an important area of financial engineering.

1.2 Quantum algorithm

The quantum computer can execute more efficient algorithms for specific problems than *classical* computers by actively utilizing the properties of quantum mechanics. It was originally conceived to solve problems in quantum mechanics efficiently. This is based on the idea that it is necessary to use a computer that explicitly employs quantum mechanics to simulate quantum states. In fact, for many-body quantum mechanical Hamiltonian,

while the resources required to simulate quantum states with a classical computer increase exponentially and make the simulation difficult to perform, a quantum computer may be feasible in the sense that the resource for the quantum computation can be reduced to polynomial.

A surprising fact is that efficient quantum algorithms exist for problems other than quantum mechanics. For example, a quantum algorithm called Shor's algorithm [3] is exponentially faster than the classical algorithm for discrete logarithm problems which is the fundamental part of modern public-key cryptography. The security of modern public-key cryptography relies on the computational complexity of the discrete logarithm problem. It is clear that the impact on the real world will be significant when a quantum computer is realized on a scale that allows Shor's algorithm to be executed. Of course, there are many other proposed applications of quantum algorithms besides quantum simulation and discrete logarithm problems. For example, quantum algorithms have been proposed for differential equations [4, 5], stochastic analysis [6], and machine learning [7]. Thus, quantum algorithms are important not only in terms of academic interest but also in industrial applications.

The ideal quantum computer has error correction capabilities and is therefore called a fault-tolerant quantum computer (FTQC). The aforementioned quantum algorithms, such as Shor's one, assume FTQC. Logical qubits must be constructed to achieve FTQC, and this requires a large number of physical qubits. While it is a very tough challenge to get large numbers of qubits to cooperate and work together, the benefits of realizing FTQC are so great that it is currently being actively developed. On the other hand, algorithms on noisy intermediate-scale quantum computers (NISQ) [8] that are not equipped with error correction have also been proposed. In particular, an algorithm called the variational quantum algorithm is not necessarily guaranteed to have provable quantum speedup, but it has been expected to be beneficial, especially in quantum chemistry [9].

In this thesis, we discuss applications of quantum algorithms, especially for financial engineering. In fact, industry demand is great because large amounts of computational resources are required to solve practical financial problems and thus, the contribution of quantum algorithms to financial engineering is expected to be significant.

It is often necessary to avoid bottlenecks in the classical-quantum transformation to achieve efficient quantum algorithms. This bottleneck appears in general applications of quantum algorithms, not only in financial engineering. Although quantum computers provide operations on Hilbert spaces of exponentially large dimensions relative to the number of qubits, it requires exponentially huge resources to embed arbitrary classical information in the space [10]. This could result in the loss of quantum speedup. Nevertheless, functions with favorable properties, such as the density function of a normal distribution, can be embedded efficiently [11]. There is also the possibility

that heuristics can be used to embed classical information efficiently, at least to an approximation [12, 13, 14, 15, 16, 17]. As for the readout of classical information from quantum information, if the target quantity is a martingale, an efficient retrieval is also possible. This thesis also discusses such efficient transformations between classical and quantum information.

1.3 Overview of thesis

Before proceeding to the description of our contributions, we introduce derivatives and stochastic analysis in Chap. 2. The fundamental theorems of asset pricing play an important role in derivative pricing. The Feynman-Kac theorem is also an important theorem that relates stochastic differential equations to partial differential equations. This chapter also includes an introduction to quantum algorithms as well as classical algorithms for derivative pricing. Regarding the classical algorithms, the Monte-Carlo method for computing the expected value and the finite difference method for simulating the partial differential equation are introduced. On the other hand, as a quantum approach, we explain the quantum amplitude estimation for the former task. For the latter task, we explain the quantum ordinary differential equation solver and the variational quantum simulation. In the chapter, we also discuss classical-quantum transformations, which are often the bottleneck in quantum algorithms.

In Chap. 3, we discuss the algorithm that simulates the probability distribution of the solution of the stochastic differential equation. This would enable the efficient transformation of classical information into quantum information. This chapter is based on [K. Kubo, Y. O. Nakagawa, S. Endo, and S. Nagayama, Physical Review A 103 (5), 052425 (2021)] and modified to fit the context.

We present the quantum algorithm that effectively performs the derivative pricing based on quantum ordinary differential solver and quantum amplitude estimation in Chap. 4. This algorithm overcomes the bottleneck of the transformation of quantum information into classical information by utilizing the fact that the present price of the derivative is estimated by the expected value in the future point of time. This chapter is based on [K. Miyamoto, K. Kubo, IEEE Transactions on Quantum Engineering 3, 3100225 (2021)] and modified to fit the context.

The algorithm introduced in Chap. 5 will be a variational version of the algorithm shown in Chap. 4. Instead of using quantum ordinary differential equation solver and quantum amplitude estimation, we use the variational quantum simulation and the SWAP test, respectively. This enables the quantum algorithm for derivative pricing to run even on small-scale quantum computers. This chapter is based on [K.Kubo, K.Miyamoto, K.Mitarai, K.Fujii, arXiv:2207.01277] and slightly modified to fit the context.

Finally, we conclude and discuss future perspectives in Chap. 6

1.4 Notation

Here, we introduce the notation used in this thesis.

- $\mathbb{R}_+$: A set of all positive real numbers.
- $\mathbb{R}_+^d$: A d-times direct product of $\mathbb{R}_+$.
- $[n]$: A set of positive integers less then or equal to n , that is, $[n] \coloneqq \{1, 2, \ldots, n\}$.
- $\boldsymbol{v}_{\wedge i}$: A vector which is made by removing an element v_i from $\boldsymbol{v} = (v_1, v_2, \ldots, v_n)^\top \in \mathbb{R}^n$, that is, $\boldsymbol{v}_{\wedge i} \coloneqq (v_1, v_2, \ldots, v_{i-1}, v_{i+1}, \ldots, v_n)$.
- $\|\boldsymbol{v}\|$: A Euclidean norm of a vector $\boldsymbol{v}$, that is, $\|\boldsymbol{v}\| = \sqrt{\sum_i v_i^2}$.
- $|i\rangle$: One of the computational basis states with a binary representation of i for an integer i.
- $|\boldsymbol{y}\rangle$: An unnormalized state where the elements of $\boldsymbol{y}$ are encoded in the amplitudes for a vector $\boldsymbol{y} = (y_1, y_2, \ldots, y_n)^\top \in \mathbb{C}^d$, that is, $|\boldsymbol{y}\rangle \coloneqq \sum_{i=1}^n y_i |i\rangle$.

Chapter 2

Preliminary

2.1 Stochastic analysis of derivative pricing

In this section, we introduce stochastic analysis for derivative pricing. We consider the derivative that refers to d underlying assets. Although prices fluctuate according to a variety of factors, we assume the underlying asset prices $\boldsymbol{S}(t) = (S_1(t), S_2(t), \ldots, S_d(t))^\top$ for $t \in \mathbb{R}_+$ evolve under stochastic processes.

Here, we introduce the stochastic differential equations. Although they are called stochastic *differential* equations, they are indeed integral equations. We define the integral equations,

$$S_i(t+s) = S(t) + \int_t^{t+s} \mu_i(u, S_i(u))du + \int_t^{t+s} \sigma_i(u, S_i(u))dW_i(u), i \in [d] \tag{2.1}$$

where μ_i, σ_i are real valued functions, and W_i are Brownian motions, which satisfy $dW_i dW_j = \rho_{ij}dt$. ρ_{ij} is an element of the correlation matrix of the underlying asset prices and satisfy $\rho_{i,i} = 1, -1 \leq \rho_{ij} \leq 1$, and $\rho_{ij} = \rho_{ji}$. The stochastic differential equations are abbreviation of Eq. (2.1) and denoted as

$$dS_i(t) = \mu_i(t, S(t))dt + \sigma_i(t, S(t))dW_i(t), i \in [d]. \tag{2.2}$$

A common model for underlying asset prices are Black-Scholes models, where the prices obey geometric Brownian motions,

$$dS_i(t) = rS_i(t)dt + \sigma_i S_i(t)dW_i(t), i \in [d] \tag{2.3}$$

where $r \in \mathbb{R}_+$ is a risk-free interest rate, $\sigma_i \in \mathbb{R}_+$ are volatility of the underlying asset prices, The risk-free interest rate generally depends on time, but, for simplicity, we assume it is constant.

In this thesis, we assume that derivatives are characterized by the payoff function $f_{\text{pay}}(\boldsymbol{S}(T))$ at maturity T and the payoff condition. For example,

a European call option is one of the simplest derivatives, and its holder has the right to buy an underlying asset at a given price $K \in \mathbb{R}$ at a given maturity T. If the underlying asset price at T is higher than K, the holder can buy the underlying asset at a price K, immediately sell it at a price $S(T)$, and then obtain $S(T) - K$. If the underlying asset price at T is lower than K, the holder can abandon the right to buy the asset and then obtain 0. Thus, the payoff function of the European call option is $f_{\text{pay}}(S(T)) = \max(S(T) - K, 0)$. *European* means that the exercise date is predetermined as T, and *call* means that the option is the right to buy. The right to sell is called *put* option. Note that there is no payoff condition for the European options. Barrier options are another type of derivative, and the payoff condition depends on the barrier level $B \in \mathbb{R}_+$. The barrier has two types; knock-out and knock-in. When the prices of underlying assets touch the knock-out barrier, the right to execute the option vanishes. When the prices of underlying assets touch the knock-in barrier, the right to execute option appears. While the derivatives introduced above refer to a single underlying asset, there also exist derivatives that refer to multiple underlying assets. Basket option is one of the multi-asset derivatives, and the payoff function is $f_{\text{pay}}(\boldsymbol{S}(T)) = \max\left(w_0 + \sum_{i\in[d]} w_i S_i(T) - K, 0\right)$, where $w_0 \in \mathbb{R}$ is a constant and $w_i \in \mathbb{R}$ are weights of the underlying assets. Asian options have the payoff determined by the average of the underlying asset prices over the predetermined periods, and thus it cannot be written in the form of a payoff function at the maturity and payoff condition. However, by introducing an additional stochastic process, it can be expressed as the payoff function only at maturity (see [2] for detail). Note that there are derivatives with more general path dependencies that we do not deal with in this thesis. Of course, multiple properties of the payoff can coexist, as in the case of barrier basket options, for example. There are a variety of derivatives other than those introduced here. Given the stochastic process of underlying assets and the payoff function of the derivative, we can evaluate the price of the derivative.

2.1.1 Fundamental theorem of asset pricing

To proceed with the derivative pricing, we need to introduce several concepts from financial engineering. First, we define *arbitrage*, which means that there is a positive probability of obtaining a profit larger than zero without a probability of loss. On the contrary, the *no-arbitrage* means that there is no arbitrage in the market. We also define *complete market*, in which the profit and loss of all derivatives are reproducible by the portfolio of the existing assets. *Martingale* is a property of a stochastic process, which means that under some measure, the expected value of the process at any point in the future is equal to the present value of the process. That is, under a probability measure Q, a martingale stochastic process $M(t), t \in [0, T]$

satisfies

$$M(0) = E_Q\left[M(t)\right], \forall t \in [0, T], \tag{2.4}$$

where E_Q is the expected value of the process under the probability measure Q. A *risk-neutral probability measure* is a probability measure under which the discounted asset price processes are martingale. With these concepts, we introduce the fundamental theorems of asset pricing.

Theorem 2.1.1 (Fundamental theorem of asset pricing I)**.** *If there exists no arbitrage in the market model, there exists a risk-neutral probability measure in the model.*

Theorem 2.1.2 (Fundamental theorem of asset pricing II)**.** *Consider the market model with a risk-neutral probability measure. This model is complete if and only if the risk-neutral measure is unique.*

The proofs of the theorems are seen in [2]. We assume that there exists no arbitrage in the market. Thus, the discounted asset prices can be a martingale under some measure from Theorem 2.1.1. We also assume that the market is complete. Then, the risk-neutral measure is unique, and the cash flows of the derivatives are reproducible by the other assets. The derivative price is equal to the value of such a portfolio. We can calculate the derivative price by evaluating the reproduced portfolio.

Consider pricing a derivative with payoff function f_{pay} at maturity T and some payoff condition. The present prices of underlying assets are given by $\boldsymbol{S}_0$. From the assumption of a complete market, there exists a portfolio that reproduces the cash flow of the derivative, and from the theorem 2.1.1, the discounted value of the portfolio is a martingale under some probability measure Q. Thus, the discounted derivative price process is also a martingale under Q, that is, the price of the derivative V satisfies

$$V(0) = E_Q\left[e^{-rt}V(t)\right], \forall t \in [0, T]. \tag{2.5}$$

At the maturity T, the derivative price $V(T)$ is equal to the payoff function $V(T) = f_{\text{pay}}\left(\boldsymbol{S}(T)\right)$. Thus, we can evaluate the derivative price by

$$V(0) = E_Q\left[e^{-rT} f_{\text{pay}}\left(\boldsymbol{S}(T)\right) \mathbb{1}_{\text{NB}} \middle| \boldsymbol{S}(0) = \boldsymbol{S}_0\right], \tag{2.6}$$

where $\mathbb{1}_{\text{NB}}$ is a random variable that takes 1 if the payoff condition is satisfied and 0 otherwise. It should be noted that $\{\boldsymbol{S}(t)\}_{t\in[0,T]}$ have to be possible paths of the underlying asset prices with the initial condition $\boldsymbol{S}(0) = \boldsymbol{S}_0$.

One way to calculate derivative prices is to use a Monte-Carlo method to generate paths under the risk-neutral probability measure and then evaluate the expected value in Eq. (2.6) with the paths. We will provide details of the Monte-Carlo method for pricing derivatives in Sec. 2.2.

2.1.2 Feynman-Kac theorem

Feynman-Kac theorem represents the relation between a stochastic differential equation and a partial differential equation. This theorem can be used to obtain a partial differential equation representing the time evolution of the expected value, i.e., the time evolution of the derivative price. Although the Monte Carlo method samples the time evolution of the underlying asset price and calculates the expected value, the derivative price can also be calculated by solving the partial differential equation.

For the derivatives whose prices satisfy Eq. (2.6), we can obtain the partial differential equation. Now, we introduce the Feynman-Kac theorem.

Theorem 2.1.3 (Feynman-Kac theorem)**.** *$\boldsymbol{S}(t)$ is stochastic processes, which satisfy the stochastic differential equation Eq. (2.2). Let $h(\boldsymbol{u})$ be a function. We define the function $g(t, \boldsymbol{s})$ as*

$$g(t, \boldsymbol{s}) = E\left[e^{-r(T-t)} h(\boldsymbol{S}(T)) \mathbb{1}_{\mathrm{NB}} \middle| \boldsymbol{S}(t) = \boldsymbol{s}\right], \tag{2.7}$$

where E is the conditional expected value conditioned by t and $\boldsymbol{s}$. Then, $g(t, \boldsymbol{s})$ satisfies the partial differential equation (PDE)

$$\frac{\partial g(t, \boldsymbol{s})}{\partial t} + \mu(t, \boldsymbol{s}) \frac{\partial g(t, \boldsymbol{s})}{\partial s_i} + \frac{1}{2} \left(\sigma(t, \boldsymbol{s})\right)^2 \frac{\partial g(t, \boldsymbol{s})}{\partial s_i \partial s_j} - r g(t, \boldsymbol{s}) = 0, \tag{2.8}$$

where $s_i \in [l_i, u_i]$ for $l_i < u_i \in \mathbb{R}_+, i \in [d]$ with boundary conditions,

$$g_i(t, \boldsymbol{s}_{\wedge i}) = g_i^{\mathrm{UB}}(t, s_1, \ldots, s_{i-1}, u_i, s_{i+1}, \ldots, s_d), \tag{2.9}$$

$$g_i(t, \boldsymbol{s}_{\wedge i}) = g_i^{\mathrm{LB}}(t, s_1, \ldots, s_{i-1}, l_i, s_{i+1}, \ldots, s_d), \tag{2.10}$$

and with the terminal condition

$$g(T, \boldsymbol{s}) = h(\boldsymbol{s}). \tag{2.11}$$

Note that we may not obtain the partial differential equation for general payoff functions with path dependence since derivative prices cannot be expressed in the form of Eq. (2.6).

2.2 Classical algorithms for derivative pricing

As described in Sec. 2.1, the present price of derivatives is equal to the expected value of the payoff function under the risk-neutral probability measure. The derivative price is also evaluated by solving the PDE. We can analytically solve the partial differential equation and obtain the closed-form solution for the simple payoff function, such as the European option. However, in general, we can not obtain the closed-form solution, and thus we

need to use numerical methods. Among the various numerical methods for calculating derivative prices, we focus on the following two approaches. One is the Monte-Carlo method, by which we simulate the stochastic process of underlying asset prices and evaluate the expected value. The other is the discretization of the partial differential equation, by which we numerically solve the resulting equation iteratively from $t = T$ and obtain the derivative price at $T = 0$.

2.2.1 Monte-Carlo method for derivative pricing

The Monte-Carlo method generally means algorithms with random numbers. In particular, it is often used to calculate expected values. We here assume the Black-Scholes (BS) model, i.e. the stochastic process of underlying assets obey Eq. (2.3). We generate the sample paths according to the stochastic differential equation. The Euler-Maruyama approximation is used as the standard method. In the Euler-Maruyama approximation, the time to maturity is divided into N_t intervals. In this case, Eq. (2.3) is approximated by the stochastic difference equation

$$S_i(t + \Delta t) - S_i(t) = rS_i(t)\Delta t + \sigma_i S_i(t) z_i \sqrt{\Delta t}, \tag{2.12}$$

where $\Delta t = T/N_t$. $\{z_i\}$ are random variables, which obey standard normal distribution. The correlation coefficient between z_i and z_j is ρ_{ij}. Starting from $t = 0$, we generate z_i and simulate Eq. (2.12) iteratively and obtain sample paths $\{\boldsymbol{S}_{l,m}\}_{l\in[N_T],m\in[N_s]}$, where $\boldsymbol{S}_{l,m}$ is underlying asset prices at $l\Delta t =: t_l$ in m-th sample under the risk neutral measure. We evaluate the expected value of Eq. (2.6) by the sample average as follows.

$$\begin{aligned} V_0 &= E_Q\left[e^{-rT} f_{\text{pay}}\left(\{\boldsymbol{S}(t)\}_{t\in[0,T]}\right)\middle|\boldsymbol{S}(0) = \boldsymbol{S}_0\right] \\ &\simeq \frac{1}{N_s} e^{-rT} \sum_{m=1}^{N_s} f_{\text{pay}}\left(\{\boldsymbol{S}_{l,m}\}_{l\in[N_T]}\right). \end{aligned} \tag{2.13}$$

We assume that the variance of f_{pay} is bounded σ^2, that is,

$$\text{Var}\left[f_{\text{pay}}\left(\{\boldsymbol{S}_{l,m}\}_{l\in[N_T]}\right)\right] \leq \sigma^2. \tag{2.14}$$

Then the probability that the error in the estimation $\hat{V}_0$ with N_s samples is ϵ away from the true price is bounded by the Chebyshev's inequality [18]

$$P\left[\left|\hat{V}_0 - V_0\right| \geq \epsilon\right] \leq \frac{\sigma^2}{k\epsilon^2} \tag{2.15}$$

Then, to estimate V_0 up to additive error ϵ with constant success probability, e.g. 99%, we need to take $N_s = O(\sigma^2/\epsilon^2)$. That is, Monte-Carlo method

requires $O(1/\epsilon^2)$ samples to obtain the solution within the accuracy ϵ. Note that the number of samples is independent of the number of assets. This is an advantage of the Monte-Carlo method. For some stochastic processes, e.g., the geometric Brownian motion, there is the closed-form solution of Eq.(2.3), and thus instead of using the Euler-Maruyama method, it is possible to sample random variables at any point in time.

2.2.2 Finite difference method for Black-Scholes partial differential equations

This subsection is based on Sec. II on [K. Miyamoto, K. Kubo, IEEE Transactions on Quantum Engineering 3, 3100225 (2021) © 2021 IEEE] and slightly modified to fit the context.

We consider the following problem.

Problem 2.2.1. *Let d be a positive integer and $T, U_1, \dots, U_d, L_1, \dots L_d$ be positive real numbers such that $L_i < U_i$ for $i \in [d]$. Define $D \coloneqq (L_1, U_1), \dots, (L_d, U_d)$, $\bar{D} \coloneqq [L_1, U_1] \times \cdots [L_d, U_d]$ and $\hat{D}^i \coloneqq [L_1, U_1] \times \cdots [L_{i-1}, U_{i-1}] \times [L_{i+1}, U_{i+1}] \times \cdots \times [L_d, U_d]$ for $i \in [d]$. Assume that a function $V : [0, T] \times \bar{D} \to \mathbb{R}$ satisfies the following PDE:*

$$\frac{\partial}{\partial t} V(t, \boldsymbol{S}) + \frac{1}{2} \sum_{i,j=1}^{d} S_i S_j \sigma_i \sigma_j \rho_{ij} \frac{\partial^2}{\partial S_i \partial S_j} V(t, \boldsymbol{S}) \tag{2.16}$$

$$+ r \left(\sum_{i=1}^{d} S_i \frac{\partial}{\partial S_i} V(t, \boldsymbol{S}) - V(t, \boldsymbol{S}) \right) = 0 \tag{2.17}$$

on $[0, T) \times D$ and boundary conditions

$$\begin{aligned} &V(T, \boldsymbol{S}) = f_{\text{pay}}(\boldsymbol{S}), \\ &V\left(t, (S_1, \dots, S_{i-1}, U_i, S_{i+1}, \dots, S_d)^T\right) = V_i^{\text{UB}}\left(t, \boldsymbol{S}_{\wedge i}\right), \text{for } i \in [d], \\ &V\left(t, (S_1, \dots, S_{i-1}, L_i, S_{i+1}, \dots, S_d)^T\right) = V_i^{\text{LB}}\left(t, \boldsymbol{S}_{\wedge i}\right), \text{for } i \in [d]. \end{aligned} \tag{2.18}$$

Here, $t \in [0, T], \boldsymbol{S} \coloneqq (S_1, \dots, S_d)^\top \in D$, $\sigma_1, \dots, \sigma_d, r$ are positive real constants such that $r < \frac{1}{2}$ for $i \in [d]$, $\rho_{ij}, i, j \in [d]$ are real constants such that $\rho_{11} = \cdots = \rho_{dd} = 1$ and the correlation matrix $\rho \coloneqq (\rho_{ij})_{\substack{1 \le i \le d \\ 1 \le j \le d}}$ is symmetric and positive definite, and $f_{\text{pay}} : D \to \mathbb{R}$, $V^{\text{UB}} : [0, T] \times \hat{D}^i \to \mathbb{R}$, and $V^{\text{LB}} : [0, T] \times \hat{D}^i \to \mathbb{R}$ are given functions. Then, for a given $\boldsymbol{S}_0 \coloneqq (S_{1,0}, \dots, S_{d,0})^\top \in D$, find $V_0 \coloneqq V(0, \boldsymbol{S}_0)$.

This partial differential equation is so-called the Black-Scholes partial differential equation (BSPDE). This is the case when the Feynman-Kac theorem introduced in Sec. 2.1.2 is applied to the BS model in Eq. (2.3). As

discussed in Sec. 2.1, the derivative price is given by the conditional expected value of the payoff discounted by the risk-free rate. That is, the price of the derivative in which the payoff $f_{\rm pay}(\boldsymbol{S}(T))$ arises at maturity T is

$$V(t,\boldsymbol{S}) = E\left[e^{-r(T-t)} f_{\rm pay}(\boldsymbol{S}(T)) \mathbb{1}_{\rm NB} \middle| \boldsymbol{S}(t) = \boldsymbol{S}\right] \tag{2.19}$$

at time t, if $\boldsymbol{S}(t) = \boldsymbol{S}$. Here, $\mathbb{1}_{\rm NB}$ is a stochastic variable taking 1 if the payoff condition (e.g., barrier condition) is satisfied or 0 otherwise. As discussed in Sec. 2.1, $V(t,\boldsymbol{S})$ satisfies Eq. (2.16) and appropriate boundary conditions.

For a later convenience, we here transform the PDE (2.16) on $[0,T)\times D$ into

$$\begin{aligned}
&\frac{\partial}{\partial\tau} Y(\tau,\boldsymbol{x}) = \mathcal{L} Y(\tau,\boldsymbol{x}) \\
&\mathcal{L} := \frac{1}{2}\sum_{i,j=1}^{d} \sigma_i\sigma_j\rho_{ij}\frac{\partial^2}{\partial x_i\partial x_j} + \sum_{i=1}^{d}\left(r - \frac{1}{2}\sigma_i^2\right)\frac{\partial}{\partial x_i},
\end{aligned} \tag{2.20}$$

on $(0,T]\times\tilde{D}$, where $\tau := T-t, \boldsymbol{x} := (x_1,...,x_d)^\top := (\log S_1,...,\log S_d)^\top$, $Y(\tau,\boldsymbol{x}) := e^{r\tau}V(T-\tau,(e^{x_1},...,e^{x_d})^\top)$, $\tilde{D} := (l_1,u_1)\times\cdots\times(l_d,u_d)$ and $u_i := \log U_i, l_i := \log L_i$ for $i\in[d]$. The boundary conditions become

$$\begin{aligned}
&Y(0,\boldsymbol{x}) = \tilde{f}_{\rm pay}(\boldsymbol{x}) := f_{\rm pay}((e^{x_1},...,e^{x_d})^\top), \\
&Y(\tau,(x_1,...,x_{i-1},u_i,x_{i+1},...,x_d)^\top) = Y_i^{\rm UB}(\tau,\boldsymbol{x}_{\wedge i}) \\
&:= V_i^{\rm UB}(T-\tau,(e^{x_1},...,e^{x_{i-1}},e^{x_{i+1}},...,e^{x_d})^\top), \text{for } i\in[d] \\
&Y(\tau,(x_1,...,x_{i-1},l_i,x_{i+1},...,x_d)^\top) = Y_i^{\rm LB}(\tau,\boldsymbol{x}_{\wedge i}) \\
&:= V_i^{\rm LB}(T-\tau,(e^{x_1},...,e^{x_{i-1}},e^{x_{i+1}},...,e^{x_d})^\top), \text{for } i\in[d]
\end{aligned} \tag{2.21}$$

The finite difference method (FDM) is a method for solving a PDE by replacing partial derivatives with finite difference approximations. In the case of (2.20), the approximation is as follows. First, letting $n_{\rm gr}$ be a positive integer, we introduce the grid points in the directions of $\boldsymbol{x}$:

$$\begin{aligned}
\boldsymbol{x}^{(k)} &:= (x_1^{(k_1)},...,x_d^{(k_d)})^\top, \\
k &= \sum_{i=1}^{d} n_{\rm gr}^{d-i}k_i + 1, k_i = 0,1,...,n_{\rm gr}-1 \\
x_i^{(k_i)} &:= l_i + (k_i+1)h_i \\
h_i &:= \frac{u_i - l_i}{n_{\rm gr}+1}.
\end{aligned} \tag{2.22}$$

Namely, there are $n_{\rm gr}$ equally spaced grid points in one direction, and the total number of the grid points in D is $N_{\rm gr} := n_{\rm gr}^d$, except ones on the boundaries. For later convenience, we set $x_i^{(-1)} = l_1$ and $x_i^{(n_{\rm gr})} = h_1$. Hereafter,

we assume that $n_{\rm gr}$ is a power of 2 for simplicity, whose detail is explained in Section 4.4, and define $m_{\rm gr} := \log_2 n_{\rm gr}$.

Then, (2.20) is transformed into the $N_{\rm gr}$-dimensional ODE system

$$\frac{d}{d\tau}\tilde{\boldsymbol{Y}}(\tau) = F\tilde{\boldsymbol{Y}}(\tau) + \boldsymbol{C}(\tau). \tag{2.23}$$

with the initial value

$$\tilde{\boldsymbol{Y}}(0) = (Y(0, \boldsymbol{x}^{(1)}), ..., Y(0, \boldsymbol{x}^{(N_{\rm gr})}))^\top = (\tilde{f}_{\rm pay}(\boldsymbol{x}^{(1)}), ..., \tilde{f}_{\rm pay}(\boldsymbol{x}^{(N_{\rm gr})}))^\top =: \tilde{\boldsymbol{f}}_{\rm pay}. \tag{2.24}$$

Here, $\tilde{\boldsymbol{Y}}(\tau), F$ and $\boldsymbol{C}(\tau)$, which newly appear in (2.23), are as follows. $\tilde{\boldsymbol{Y}}(\tau) := (\tilde{Y}_1(\tau), ..., \tilde{Y}_{N_{\rm gr}}(\tau)) \in \mathbb{R}^{N_{\rm gr}}$ and its k-th element is an approximation of $Y(\tau, x^{(k)})$. F is a $N_{\rm gr} \times N_{\rm gr}$ real matrix, which is expressed by a sum of Kronecker products of $n_{\rm gr} \times n_{\rm gr}$ matrices, that is,

$$\begin{aligned}
F &:= F^{\rm 2nd} + F^{\rm 1st} \\
F^{\rm 2nd} &:= \sum_{i=1}^{d} \frac{\sigma_i^2}{2h_i^2} I^{\otimes i-1} \otimes D^{\rm 2nd} \otimes I^{\otimes d-i} \\
&\quad + \sum_{i=1}^{d-1} \sum_{j=i+1}^{d} \frac{\sigma_i \sigma_j \rho_{ij}}{4h_i h_j} I^{\otimes i-1} \otimes D^{\rm 1st} \otimes I^{\otimes j-i-1} \otimes D^{\rm 1st} \otimes I^{\otimes d-j} \\
F^{\rm 1st} &:= \sum_{i=1}^{d} \frac{1}{2h_i} \left(r - \frac{1}{2}\sigma_i^2 \right) I^{\otimes i-1} \otimes D^{\rm 1st} \otimes I^{\otimes d-i},
\end{aligned} \tag{2.25}$$

where I is the $n_{\rm gr} \times n_{\rm gr}$ identity matrix and

$$D^{\rm 1st} := \begin{pmatrix} 0 & 1 & & & & \\ -1 & 0 & 1 & & & \\ & -1 & 0 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & -1 & 0 & 1 \\ & & & & -1 & 0 \end{pmatrix}, D^{\rm 2nd} := \begin{pmatrix} -2 & 1 & & & & \\ 1 & -2 & 1 & & & \\ & 1 & -2 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & -2 & 1 \\ & & & & 1 & -2 \end{pmatrix} \tag{2.26}$$

are $n_{\rm gr} \times n_{\rm gr}$ tridiagonal matrices. $\boldsymbol{C}(\tau) := (C_1(\tau), ..., C_{N_{\rm gr}}(\tau))^\top$ is necessary

to take into account the boundary conditions and its k-th element is

$$\begin{aligned} C_k(\tau) &= \sum_{i=1}^{d} \frac{\sigma_i^2}{2h_i^2} \left[\delta_{k_i,0} Y_i^{\mathrm{LB}}(\tau, \boldsymbol{x}_{\wedge i}^{(k)}) + \delta_{k_i,n_{\mathrm{gr}}-1} Y_i^{\mathrm{UB}}(\tau, \boldsymbol{x}_{\wedge i}^{(k)}) \right] \\ &+ \sum_{i=1}^{d-1} \sum_{j=i+1}^{d} \frac{\sigma_i \sigma_j \rho_{ij}}{4h_i h_j} \left[-\delta_{k_i,0} Y_i^{\mathrm{LB}}(\tau, \boldsymbol{x}_{\wedge i}^{(k)}) - \delta_{k_j,0} Y_j^{\mathrm{LB}}(\tau, \boldsymbol{x}^{(k)})_{\wedge i} \right. \\ &+ \left. \delta_{k_i,n_{\mathrm{gr}}-1} Y_i^{\mathrm{UB}}(\tau, \boldsymbol{x}_{\wedge i}^{(k)}) + \delta_{k_j,n_{\mathrm{gr}}-1} Y_j^{\mathrm{UB}}(\tau, \boldsymbol{x}_{\wedge i}^{(k)}) \right] \\ &+ \sum_{i=1}^{d} \frac{1}{2h_i} \left(r - \frac{1}{2}\sigma_i^2 \right) \left[\delta_{k_i,n_{\mathrm{gr}}-1} Y_i^{\mathrm{UB}}(\tau, \boldsymbol{x}_{\wedge i}^{(k)}) - \delta_{k_i,0} Y_i^{\mathrm{LB}}(\tau, \boldsymbol{x}_{\wedge i}^{(k)}) \right] \\ &. \end{aligned} \tag{2.27}$$

Then, let us discuss the accuracy of the approximation (2.23). First, we make a following assumption.

Assumption 2.2.1. *$Y(\tau, \boldsymbol{x})$, the solution of (2.20) and (2.21), is four-times differentiable with respect to $x_1, ..., x_d$ and there exist $\zeta, \xi \in \mathbb{R}$ such that*

$$\forall i,j,k,l \in [d], \tau \in (0,T), \boldsymbol{x} \in \tilde{D}, \left| \frac{\partial^3 Y}{\partial x_i \partial x_j \partial x_k}(\tau, \boldsymbol{x}) \right| < \zeta, \left| \frac{\partial^4 Y}{\partial x_i \partial x_j \partial x_k \partial x_l}(\tau, \boldsymbol{x}) \right| < \xi. \tag{2.28}$$

We then obtain the following lemma, as proved in Appendix A2.1.

Lemma 2.2.1. *Let $Y(\tau, \boldsymbol{x})$ be the solution of (2.20) and (2.21), and $\tilde{\boldsymbol{Y}}(\tau)$ be that of (2.23) and (2.24). Under Assumption 2.2.1, if, for a given $\epsilon \in \mathbb{R}_+$,*

$$h_i < \min \left\{ \frac{1}{d\sigma_i} \sqrt{\frac{3\epsilon}{2\xi T}}, \frac{1}{\sigma_i} \sqrt{\frac{3\epsilon}{\zeta d T}} \right\}, i \in [d] \tag{2.29}$$

then, for any $\tau \in (0,T)$, the inequality

$$\| \tilde{\boldsymbol{Y}}(\tau) - \boldsymbol{Y}(\tau) \| < \sqrt{N_{\mathrm{gr}}} \epsilon \tag{2.30}$$

holds, where $\boldsymbol{Y}(\tau) = (Y(\tau, \boldsymbol{x}^{(1)}), ..., Y(\tau, \boldsymbol{x}^{(N_{\mathrm{gr}})}))^\top$.

Lemma 2.2.1 means that the root mean square of the differences between $\tilde{Y}_i(\tau)$ and $Y(\tau, \boldsymbol{x}^{(i)})$ is upper bounded by ϵ.

2.3 Quantum algorithms for derivative pricing

In this section, we introduce derivative price calculation using quantum algorithms. As explained in Sec. 2.2, there are two main approaches to derivatives price calculation: the Monte-Carlo approach and the PDE approach.

In the classical Monte-Carlo method, a number of sampling times of $O(1/\epsilon^2)$ is needed to calculate the derivative price with an accuracy ϵ. On the other hand, a quantum algorithm called quantum amplitude estimation can obtain a solution with precision ϵ with a number of measurements of $O(1/\epsilon)$. In this sense, it is described as a quadratic speedup. In the classical approach using PDEs, as the number of assets increases, computational complexity increase exponentially. When the finite difference method is used in the quantum algorithm, the number of qubits needed to simulate a linear PDE is $O(\mathrm{polylog}(N_{\mathrm{gr}}))$, so in this sense, the computational complexity is reduced exponentially.

2.3.1 Quantum amplitude estimation

In Sec. 2.2.1, we introduced the derivative price using the classical Monte-Carlo method and showed that its error decreases with the square root of the number of samples. We here show the way to estimate the price of the derivative using the quantum amplitude estimation introduced in Ref. [19]. For simplicity, we consider the single-asset European call option.

Recall that the present price of the European call option is the expected value of the payoff function at maturity T, which is given by

$$V = e^{-rT} E\left[\max\left(S(T) - K, 0\right) | S(0) = S_0\right]. \tag{2.31}$$

The expected value is approximated as

$$\begin{aligned} &E\left[\max\left(S(T) - K, 0\right) | S(0) = S_0\right] \\ &\simeq \sum_{x=0}^{2^n-1} p(S(T) = x; S(0) = S_0) \max\left(x - K, 0\right), \end{aligned} \tag{2.32}$$

where $p(S(T) = x; S(0) = S_0)$ is the probability that $S(T)$ takes x conditioned on $S(0) = S_0$. We consider the way to calculate this value with quantum algorithm. We assume that there exists a unitary operator $\mathcal{G}$ which satisfy

$$\mathcal{G}\left|0\right\rangle = \sum_{j=0}^{2^n-1} \sqrt{p(x_j)}\left|j\right\rangle, \tag{2.33}$$

where $p(x_j) = p(S(T) = x_j; S(0) = S_0)$. For the BS model, $\mathcal{G}$ is constructed by Grover-Rudolph method [11]. We define a binary approximation of $\max(x - K, 0)$ as $v(x_j) : \{0,1\}^n \to \{0,1\}^n$ and assume that there exists $\mathcal{R}$ which acts as

$$\mathcal{R}\left|j\right\rangle\left|0\right\rangle = \left|j\right\rangle\left(\sqrt{1 - v(x_j)}\left|0\right\rangle + \sqrt{v(x_j)}\left|1\right\rangle\right). \tag{2.34}$$

By using $\mathcal{G}$ and $\mathcal{R}$, we can implement an operator $\mathcal{A}$, which acts on a reference state with $n+1$ qubits and yields

$$\mathcal{A}(|0\rangle_{n+1}) = \sum_{j=0}^{2^n-1} \sqrt{p(x_j)}\,|j\rangle \left(\sqrt{1-v(x_j)}\,|0\rangle + \sqrt{v(x_j)}\,|1\rangle\right) =: |\chi\rangle\,. \quad (2.35)$$

Here, we define an operator

$$\mathcal{S}_1 := I^{\otimes n+1} - 2I^{\otimes n} \otimes |1\rangle\langle 1|\,. \quad (2.36)$$

Since the any quantum state in the $(n+1)$-qubit Hilbert space can be expressed by the linear combination of $|\chi\rangle$ and its orthogonal state $\left|\chi^{\perp}\right\rangle$, we can express

$$\mathcal{S}_1 |\chi\rangle = \cos(\theta/2)\,|\chi\rangle + e^{i\phi}\sin(\theta/2)\left|\chi^{\perp}\right\rangle, \quad (2.37)$$

where $\theta, \phi \in [0, 2\pi]$. Note that a measurement of $\mathcal{S}_1$ on $|\chi\rangle$ yields

$$1 - 2\sum_{j=0}^{2^m-1} p(x_j)v(x_j) = \cos(\theta/2). \quad (2.38)$$

Thus, we can calculate the expected value $\sum_{j=0}^{2^m-1} p(x_j)v(x_j)$, i.e., the present price of the derivative by estimating θ. By constructing an operator Q whose eigenvalues $\pm\theta$, we can estimate θ by the quantum phase estimation. To this end, we define $\mathcal{S}_\chi := I^{\otimes n+1} - 2|\chi\rangle\langle\chi|$. Note that $-\mathcal{S}_\chi$ reflects across $|\chi\rangle$. $\mathcal{S}_\chi$ can be written as

$$\begin{aligned}\mathcal{S}_\chi &= \mathcal{A}\left(I^{\otimes n+1} - 2|0\rangle_{n+1}\langle 0|_{n+1}\right)\mathcal{A}^\dagger \\ &= \mathcal{A}\mathcal{Z}\mathcal{A}^\dagger,\end{aligned} \quad (2.39)$$

where $\mathcal{Z} := I^{\otimes n+1} - 2|0\rangle_{n+1}\langle 0|_{n+1}$ is a reflection on the computational basis. We also define an operator

$$\mathcal{S}_{\psi_\chi} := I^{\otimes n+1} - 2\mathcal{S}_1|\chi\rangle\langle\chi|\mathcal{S}_1 \quad (2.40)$$

Note that $-\mathcal{S}_{\psi_\chi}$ reflects across $\mathcal{S}_1|\chi\rangle$. The operator

$$Q := \mathcal{S}_{\psi_\chi}\mathcal{S}_1 \quad (2.41)$$

is a rotation by an angle 2θ on the plane, and its eigenvalues are $e^{\pm i\theta}$. By using quantum phase estimation, we can calculate θ and then, obtain the expected value by Eq. (2.38). To estimate the present price of the derivative within the error ϵ, the total number of applications of $\mathcal{S}_{\psi_\chi}$ is

$$\tilde{O}(\lambda/\epsilon) \quad (2.42)$$

where $\tilde{O}$ ignores the polylogarithmic factors, and λ is square root of the upper bound of the variance of the f_{pay} defined in Eq. (2.13). For more detail, see Ref. [19].

2.3.2 Quantum algorithm for solving ordinary differential equation systems

In this subsection, we outline the algorithm of [4]. This subsection is based on Sec. III in [K. Miyamoto, K. Kubo, IEEE Transactions on Quan- tum Engineering 3, 3100225 (2021) © 2021 IEEE] and slightly modified fitting in the context. This is the algorithm for solving the linear ODE system

$$\frac{d}{dt}\boldsymbol{x}(t) = A\boldsymbol{x}(t) + \boldsymbol{b}, \tag{2.43}$$

with the initial condition $\boldsymbol{x}(0) = \boldsymbol{x}_{\text{ini}}$. Here, $\boldsymbol{x}(t)$ and $\boldsymbol{b}$ are N-dimensional real-valued vectors and $A \in \mathbb{R}^{N\times N}$ is a constant diagonalizable matrix. Suppose that we want to find $\boldsymbol{x}(T)$ for some $T \in \mathbb{R}_+$. The algorithm is based on the formal solution of (2.43)

$$\boldsymbol{x}(T) = e^{AT}\boldsymbol{x}_{\text{ini}} + (e^{AT} - I_N)A^{-1}\boldsymbol{b}. \tag{2.44}$$

In order to calculate $\boldsymbol{x}(t)$, we consider the linear equation system on the tensor product space $V := \mathbb{R}^{q+1} \otimes \mathbb{R}^N$, where $\mathbb{R}^{q+1}$ corresponds to the auxiliary space and $\mathbb{R}^N$ corresponds to the original space on which A operates:

$$C_{m,k,p}(Ah_t)\boldsymbol{X} = \boldsymbol{e}_0 \otimes \boldsymbol{x}_{\text{ini}} + h\sum_{i=0}^{m-1} \boldsymbol{e}_{i(k+1)+1} \otimes \boldsymbol{b}. \tag{2.45}$$

Here, m, p, k are positive integers set large enough (see the statement of Theorem 4.2.1), $q := m(k+1)+p$, $h_t = T/m$, $\boldsymbol{X} \in \mathbb{R}^{N(q+1)}$ and $\{\boldsymbol{e}_i\}_{i=0,1,\ldots,q}$ is an orthonormal basis of $\mathbb{R}^{q+1}$. For $B \in \mathbb{R}^{N\times N}$, the $N(q+1) \times N(q+1)$ matrix $C_{m,k,p}(B)$ is defined as

$$\begin{aligned} C_{m,k,p}(B) \;:=\; & \sum_{j=0}^{q} \boldsymbol{e}_j\boldsymbol{e}_j^\top \otimes I_N - \sum_{i=0}^{m-1}\sum_{j=1}^{k} \boldsymbol{e}_{i(k+1)+j}\boldsymbol{e}_{i(k+1)+j-1}^\top \otimes \frac{1}{j}B \\ & - \sum_{i=0}^{m-1}\sum_{j=0}^{k} \boldsymbol{e}_{(i+1)(k+1)}\boldsymbol{e}_{i(k+1)+j}^\top \otimes I_N - \sum_{j=m(k+1)+1}^{q} \boldsymbol{e}_j\boldsymbol{e}_{j-1}^\top \otimes I. \end{aligned} \tag{2.46}$$

Visually, (2.45) is displayed as follows

$$
\begin{pmatrix}
I_N & & & & & & & & & & & & \\
-Ah_t/1 & I_N & & & & & & & & & & & \\
 & \ddots & \ddots & & & & & & & & & & \\
 & & -Ah_t/k & I_N & & & & & & & & & \\
-I_N & \cdots & -I_N & -I_N & I_N & & & & & & & & \\
 & & & & \ddots & \ddots & & & & & & & \\
 & & & & & -Ah_t/1 & I_N & & & & & & \\
 & & & & & & \ddots & \ddots & & & & & \\
 & & & & & & & -Ah_t/k & I_N & & & & \\
 & & & & & -I_N & \cdots & -I_N & -I_N & I_N & & & \\
 & & & & & & & & & -I_N & I_N & & \\
 & & & & & & & & & & \ddots & \ddots & \\
 & & & & & & & & & & & -I_N & I_N
\end{pmatrix}
\boldsymbol{X} =
\begin{pmatrix}
\boldsymbol{x}_{\mathrm{ini}} \\ h_t\boldsymbol{b} \\ 0 \\ \vdots \\ 0 \\ \vdots \\ h_t\boldsymbol{b} \\ 0 \\ \vdots \\ 0 \\ 0 \\ \vdots \\ 0
\end{pmatrix}. \tag{2.47}
$$

$C_{m,k,p}$ is designed based on the Taylor expansion of (2.44). The solution of (2.45) can be written as

$$
\boldsymbol{X} = \sum_{i=0}^{m-1} \sum_{j=1}^{k} \boldsymbol{e}_{i(k+1)+j} \otimes \boldsymbol{x}_{i,j} + \sum_{j=0}^{p} \boldsymbol{e}_{m(k+1)+j} \otimes \boldsymbol{x}_m, \tag{2.48}
$$

for some vectors $\boldsymbol{x}_{i,j}, \boldsymbol{x}_m \in \mathbb{R}^N$, and $\boldsymbol{x}_m$ becomes close to $\boldsymbol{x}(T)$, which we want to find. Note that $\boldsymbol{x}_m$ is repeated p times in the solution $\boldsymbol{X}$, which enhances the probability of obtaining the desired vector in the output quantum state of the algorithm.

Although the $C_{m,k,p}(Ah_t)$ is an extremely large matrix, the quantum algorithms for solving linear equation systems (QLS algorithms)[20, 21, 22, 23] can output the solution of (2.45) only with complexity of $O(\log \mathcal{N})$, where $\mathcal{N}$ is the number of rows (or columns) in $C_{m,k,p}(Ah_t)$.

2.3.3 Variational quantum simulation

In this subsection, we introduce the VQS, which is a variational quantum algorithm to solve linear ODEs [5, 24, 9]. This subsection is based on Sec. II.B in [Kubo, Miyamoto, Mitarai, and Fujii, arXiv:2207.01277] and slightly modified fitting in the context. Consider solving the following linear ODE,

$$
\frac{d}{dt}\boldsymbol{v}(t) = L(t)\boldsymbol{v}(t) + \boldsymbol{u}(t), \boldsymbol{v}(0) = \boldsymbol{v}_0. \tag{2.49}
$$

where $\boldsymbol{v}(t) = (v_1(t), \ldots, v_{N_v}(t))$, $\boldsymbol{v}_0 = (v_{0,1}, \ldots, v_{0,N_v})$, $\boldsymbol{u}(t) = (u_1(t), \ldots, u_{N_v}(t)) \in \mathbb{C}^{N_v}$, and $L(t)$ is an (possibly non-hermitian) operator. To simulate the vector $\boldsymbol{v}(t)$, we instead simulate an unnormalized quantum state $|\boldsymbol{v}(t)\rangle$, which is the solution of

$$
\frac{d}{dt}|\boldsymbol{v}(t)\rangle = L(t)|\boldsymbol{v}(t)\rangle + |\boldsymbol{u}(t)\rangle, |\boldsymbol{v}(0)\rangle = |\boldsymbol{v}_0\rangle. \tag{2.50}
$$

Here, we make three assumptions. First, $L(t)$ can be decomposed as

$$L(t) = \sum_{k=1}^{N_L} \lambda_k(t) U_k^L(t), \tag{2.51}$$

where $\lambda_k(t)$ is real, and $U_k^L(t)$ are quantum gates. Second, $|\boldsymbol{u}(t)\rangle$ can be written as

$$|\boldsymbol{u}(t)\rangle = \sum_{l=1}^{N_u} \eta_l(t) U_l^u(t) |0\rangle , \tag{2.52}$$

where $\eta_l(t)$ is real, and $U_l^u(t)$ are quantum gates. Third, there are some constant $\alpha_v \in \mathbb{C}$ and an quantum gate U_v such that $|\boldsymbol{v}_0\rangle = \alpha_v U_v |0\rangle$. In VQS, we approximate $|\boldsymbol{v}(t)\rangle$ by an unnormalized ansatz state $|\tilde{v}(\boldsymbol{\theta}(t))\rangle \coloneqq \theta_0(t) R_1(\theta_1(t)) R_2(\theta_2(t)) \cdots R_{N_a}(\theta_{N_a}(t)) |\boldsymbol{v}_0\rangle$ and determine parameters $\boldsymbol{\theta}(t) = (\theta_0(t), \theta_1(t), \ldots, \theta_{N_a}(t))^\top \in \mathbb{R}^{N_a+1}$ by the variational principle. Here, $R_k(\theta_k) = W_k e^{i\theta_k G_k}$ are parameterized quantum circuits, W_k are quantum gates, and $G_k \in \{X, Y, Z, I\}^{\otimes n}$ are multi-qubit Pauli gates with n-qubit system. By McLachlan's variational principle [25]

$$\min_{\boldsymbol{\theta}} \left\| \frac{d}{dt} |\tilde{v}(\boldsymbol{\theta}(t))\rangle - L(t) |\tilde{v}(\boldsymbol{\theta}(t))\rangle - |\boldsymbol{u}(t)\rangle \right\|^2 , \tag{2.53}$$

we obtain the differential equation [5]

$$\sum_{n=0}^{N_a} \mathcal{M}_{m,n} \dot{\theta}_n(t) = \mathcal{V}_m, \tag{2.54}$$

where

$$\mathcal{M}_{i,j} = \mathrm{Re}\left(\frac{\partial \langle \tilde{v}(\boldsymbol{\theta}(t))|}{\partial \theta_i} \frac{\partial |\tilde{v}(\boldsymbol{\theta}(t))\rangle}{\partial \theta_j} \right), \tag{2.55}$$

$$\mathcal{V}_j = \sum_{k=1}^{N_L} \lambda_k(t) \, \mathrm{Re}\left(\frac{\partial \langle \tilde{v}(\boldsymbol{\theta}(t))|}{\partial \theta_j} U_k^L(t) |\tilde{v}(\boldsymbol{\theta}(t))\rangle \right) + \sum_{l=1}^{N_u} \eta_l(t) \, \mathrm{Re}\left(\frac{\partial \langle \tilde{v}(\boldsymbol{\theta}(t))|}{\partial \theta_n} U_l^u(t) |0\rangle \right). \tag{2.56}$$

We can evaluate each term in Eqs. (2.55)(2.56) by quantum circuits presented in Appendix A3.4. Then, we solve Eq. (2.54) classically and obtain $\dot{\theta}_j(t)$. Note that the number of measurements needed to evaluate $\mathcal{M}_{i,j}$ and $\mathcal{V}_i$ by the Hadamard test within the accuracy $\bar{\epsilon}$ is $O(|\theta_0(t)|^2/\bar{\epsilon}^2)$. This is because $\mathcal{M}_{i,j}$ and $\mathcal{V}_i$ contain the normalization factor $\theta_0(t)$ when $i > 0$ or $j > 0$ (see Appendix A3.4). We assume that $|\theta_0(t)|$ is upper-bounded by

some constant. In the simulation of BSPDE, $|\theta_0(t)|^2$ is about a ratio of the sum of the squares of the derivative prices at time $T-t$ to the sum of the squares of the payoff function at maturity. Since the derivative price is the expected value of the payoff function, this assumption is satisfied if the value range of the payoff function is finite. Starting from $t = 0$, we obtain the time evolution of $\boldsymbol{\theta}(t)$ by repeating

$$\boldsymbol{\theta}(t + \Delta t) \leftarrow \boldsymbol{\theta}(t) + \dot{\boldsymbol{\theta}}(t)\Delta t, \tag{2.57}$$

where Δt is an interval in time direction. Consequently, we obtain $|\tilde{v}(\boldsymbol{\theta}(t))\rangle$ which approximates $|\boldsymbol{v}(t)\rangle$.

Therefore, to ensure a feasible VQS algorithm, both $M, k_{\text{term}}(t)$, and the depth of the unitaries U_k must be $O(\text{poly}(n))$.

2.4 Classical-Quantum transformation

As described later, the quantum algorithms for derivative pricing introduced above all require the embedding of probability distributions. In addition, the quantum algorithms for solving differential equations introduced in Secs. 2.3.2 and 2.3.3 require the retrieval of classical information from the quantum state since the present price of the derivative is embedded in the quantum state. In this section, we discuss embedding classical probability distribution into the quantum state and retrieving classical information from the quantum state. If these inputs and outputs are not efficient, quantum speedup will be lost.

There are several previous studies on how to embed classical functions into quantum states. Ref. [11] has shown a method for embedding functions that can efficiently compute integrals in classical calculations into the amplitude of a quantum state. Specifically, given a log-concave function $p(x)$, they construct a unitary operator that generates

$$|\phi\rangle = \sum_{i=0}^{2^n-1} \sqrt{p_i}\,|i\rangle\,. \tag{2.58}$$

where p_i is a discretized version of $p(x)$. Moreover, the algorithm shown in Ref. [26] can embed arbitrary continuous functions by using the adiabatic theorem and Hamiltonian simulation. Several algorithms have been proposed for small-scale quantum computers that use heuristics to embed functions into quantum states. Popular examples are the quantum generative adversarial network (qGAN) [12, 13, 14, 15] and the quantum circuit Born machine (QCBM) [16]. The qGAN consists of a generator composed of a variational quantum circuit and a classical neural network. By interplaying the generator and the discriminator, qGAN learns a generator that generates the desired quantum state. In QCBM, a similar generative model can

be obtained by optimizing the loss function called the kernelled maximum mean discrepancy loss. In these algorithms, we optimize the parameters of a parameterized circuit $\mathcal{U}(\boldsymbol{\theta})$ which generate $|\phi(\boldsymbol{\theta})\rangle \coloneqq \mathcal{U}(\boldsymbol{\theta})\,|0\rangle$ such that

$$|\,\langle\phi(\boldsymbol{\theta})|i\rangle\,|^2 = p_i, i \in [0, 2^n - 1]. \tag{2.59}$$

Although the method using heuristics has no guarantee of computational complexity, it has the potential to be implemented with shallow quantum circuits compared to the algorithm in Refs. [11, 26]. These algorithms can be used for state preparation in derivative pricing. However, for path-dependent derivatives, probability distributions at multiple points in time may be needed. In that case, it may be necessary to run the algorithms above for the number of points in the time needed.

On the other hand, to extract a single amplitude from a quantum state, the Born rule requires a computational complexity that is exponential with respect to the number of qubits. In fact, Refs. [27, 28, 29, 30] show the way to obtain a quantum state in which derivative prices are embedded, but this extraction is a bottleneck. However, as explained in Chap. 4, 5, this problem can be avoided by utilizing the fact that the derivative price is a martingale.

Chapter 3

Variational quantum simulation of the stochastic differential equation

In this chapter we present a variational quantum algorithm to simulate the probability distribution of a Markovian stochastic process. As discussed in Secs. 2.3.1 and 2.3.2, we have to embed the probability distribution into a quantum state to calculate the expected value. In specific cases such as Brownian motions, the embedding is performed efficiently [11, 31]. However, in general cases, the embedding is difficult and would lose the quantum speedup. To overcome this problem, we simulate the time evolution of the probability distribution derived from stochastic differential equation using the variational quantum simulation (VQS). VQS approximates the time evolution of an unnormalized quantum state by the time evolution of low-dimensional parameters. We can approximate the probability distribution of a stochastic process at a desired point in time by using this time evolution. Therefore, this method can be considered as one of the quantum generative models using variational quantum computation. Unlike other quantum generative models, it has the advantage that the optimization does not need to be computed in the algorithm. Furthermore, once the algorithm is executed, we obtain quantum circuits that generate probability distributions at multiple times. This can be a useful property, especially in pricing path-dependent derivatives. This chapter is based on [K. Kubo, Y. O. Nakagawa, S. Endo, and S. Nagayama, Physical Review A 103 (5), 052425 (2021)] with modifications for fitting in the context.

3.1 Introduction

Stochastic differential equations (SDEs), which describe the time evolution of random variables, are among the most important mathematical tools for

modeling uncertain systems in diverse fields, such as finance [2], physics [32], and biology [33]. From the expectation values of the simulated random variables, we can often extract information about the system of interest. Since the expectation values rarely admit analytical solutions, they are usually obtained by numerical methods such as the Monte Carlo method [34]. However, those numerical methods incur high computational costs, especially in high-dimensional problems such as the SDEs of financial applications [35, 36, 37]. Therefore, a method that can speed-up SDE simulations is urgently demanded.

Such a speed up can be achieved on quantum computers. Throughout the past decade, technological developments have realized a primitive form of quantum computers called noisy intermediate-scale quantum (NISQ) devices [38], which can handle problems that are intractably large for classical computers [39]. NISQ devices can operate only a few tens to hundreds of qubits without error correction, so they cannot run quantum algorithms requiring deep and complicated quantum circuits. Although quantum algorithms are expected to outperform classical ones on specific computing tasks [40, 3, 20, 41], they usually exceed the capability of NISQ devices. Accordingly, NISQ devices have been leveraged with heuristic algorithms that solve real-world problems. For example, in quantum chemistry and condensed matter physics, the variational quantum eigensolver (VQE) algorithm [42, 43] can calculate the ground-state energies of given Hamiltonians [44, 45]. Another example is quantum machine learning with variational quantum circuits [46, 47, 48, 49]. Both algorithms variationally optimize the tuneable classical parameters in quantum circuits, so the speedups of the computation over classical computers and the precision of the obtained results are not guaranteed in general.

Several quantum-computing-based methods obtain the expectation value of a function that takes an SDE solution as its argument. However, all of these methods require prerequisite knowledge of the SDE solution. In [29], the partial differential equation describing the time evolution of the expectation value was simulated by a variational quantum computation, which requires pre-derivation of the partial differential equation of the expectation value. In [19] and [15], the probability distribution of the SDE solution was embedded in the quantum state, and the expectation value was calculated by a quantum amplitude estimation algorithm (QAE). In this case, it is needed to be able to sample from the probability distribution of the SDE solutions.

In this chapter, to solve an SDE with quantum algorithms, we apply a tree model approximation [50], and hence obtain a linear differential equation describing the probability distribution of SDE solutions. This differential equation is then solved by a variational quantum simulation (VQS) [51, 52, 24, 5, 53]. Note that linear differential equations can be solved by a quantum linear solver algorithm (QLSA) [20, 54, 55, 4], which

is expected to be quantum-accelerated. However, the QLSA requires a large number of ancilla qubits and deep circuits and is possibly executable only on quantum computers with error correction. Our proposed method possesses several desirable features. First, the probability distribution is simulated by the tree-model approximation, so the model requires only the SDE. Our method may be more efficient because it does not require sampling from the probability distribution of SDE solutions. Second, once the VQS is performed, the variational parameters are obtained as classical information, and the probability distribution of the simulation results can be used to compute various expectation values. We can also compute path-dependent expectation values because the time series of the probability distribution is obtained. Third, the algorithm is less resource-intensive than the QLSA. Since VQS is a variational algorithm, it is difficult to estimate the exact computational cost, but VQS requires only a few ancilla qubits and calculates the expectation value for relatively shallow unitary gates at each time step. The number of qubits and the depth of the circuit are expected to be much smaller than QLSA. As our method uses a new scheme for embedding probability distributions in quantum states, the method for computing expectation values is also new. We additionally found that the expectation values are more simply determined by our method than by the QAE. The proposed method facilitates the application of SDEs in quantum computing simulations and is expected to impact various scientific fields.

The remainder of this chapter is organized as follows. Section 3.2 reviews the trinomial tree-model approximation and the VQS, before introducing our method. Our main theoretical results are contained in Secs. 3.3, 3.4. Section 3.3 proposes a VQS-based method that simulates the dynamics of the probability distribution of the stochastic process in the trinomial tree model. The quantum circuits and operators that perform the VQS are also constructed in this section. Section 3.4 calculates the expectation value of the random variable using the state obtained by simulating SDE with the VQS. Section 3.5 discusses the advantages of our method and compares them with previous studies. Section 3.6 numerically evaluates our algorithm on two SDE prototypes: the geometric Brownian motion and the Ornstein-Uhlenbeck process. Conclusions are presented in Section 3.7. Appendix A1.1 analyses the complexity of calculating the expectation value, and Appendix A1 generalizes our result to a multiple-variable process. Appendix A1.5 evaluates the error of expectation values from piecewise polynomial approximation.

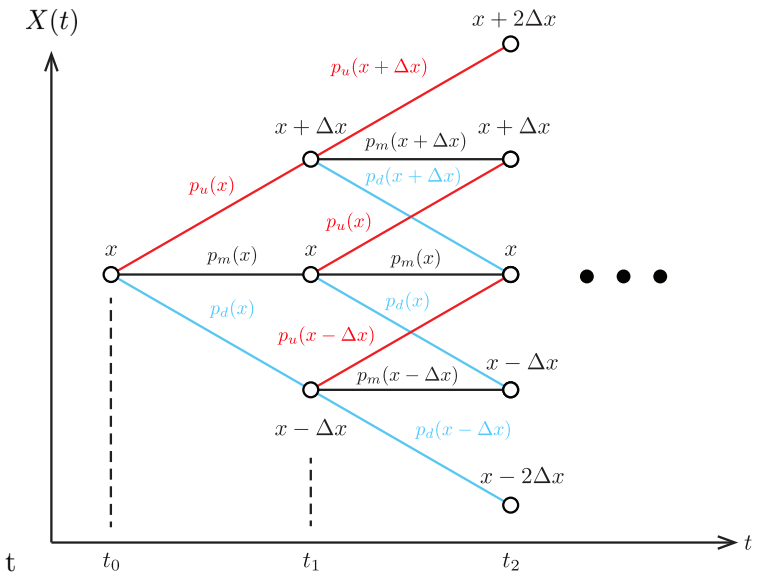


Figure 3.1: Lattice of the trinomial tree model. Nodes (circles) at (t, x) represent the events in which $X(t)$ takes the value x. Edges represent the transition probabilities between the nodes. The stochastic process starts at node (t_0, x_0) and "hops" to the other nodes depending on the transition probabilities.

3.2 Trinomial tree-model approximation of the stochastic differential equation

This section reviews the trinomial tree-model approximation of the SDE [50]. In Sec. 3.3, we combine the trinomial tree-model and VQS into a method that simulates the SDE by the VQS.

Let us consider a random variable $X(t)$ taking values on an interval $I \subset \mathbb{R}$. We refer to I as an event space. The SDE of a single process $\{X(t)\}_{t\in[0,T]}$, which is a time-series of random variables from $t = 0$ to $t = T$, is defined as [2]

$$dX(t) = \mu(X(t), t)dt + \sigma(X(t), t)dW,\ X(0) = x_{\text{ini}} \in I, \tag{3.1}$$

where $\mu(X(t), t), \sigma(X(t), t)$ are real valued functions of time t and the variable $X(t)$, and W denotes the Brownian motion. In the main text, our proposal is applied to a single process (extensions to multi-variables cases are described in Appendix A1).

The tree model numerically simulates the time evolution of an SDE. Let us consider an SDE simulation of the process with event space $[0, x_{\max}]$ from $t = 0$ to $t = T$. We discretize the time as $t_i \equiv i\Delta t$ $(i = 0, 1, \ldots, N_t)$ and the event space as $x_i \equiv i\Delta x$ $(i = 0, 1, \ldots, N_x)$, where $N_t\Delta t = T$ and $N_x\Delta x =$

$x_{\max}$. In this discretization scheme, we define a $(N_x+1)\times(N_t+1)$ lattice on which each node (i,j) is associated with a probability $\mathrm{Prob}[X(t_j)=x_i]$ and each edge represents a transition between two nodes, as shown in Fig. 3.1. Here, we adopt the trinomial tree model, which has three transition probabilities as follows:

$$\begin{aligned} p_u(x,t) &= \mathrm{Prob}[X(t+\Delta t)=x+\Delta x \,|\, X(t)=x], \\ p_d(x,t) &= \mathrm{Prob}[X(t+\Delta t)=x-\Delta x \,|\, X(t)=x], \\ p_m(x,t) &= \mathrm{Prob}[X(t+\Delta t)=x \,|\, X(t)=x]. \end{aligned}$$

These probabilities were chosen to reproduce the first and second moment (mean and variance, respectively) of the random variable $X(t)$ in Eq. (3.1). Following the Euler-Maruyama method [37], the SDE is discretized as

$$X(t_{j+1})-X(t_j)=\mu(X(t_j),t)\Delta t+\sigma(X(t_j),t)\sqrt{\Delta t}z, \tag{3.2}$$

where $z\sim N(0,1)$ and $O(\Delta t^2)$ terms are ignored. The conditional expectation value and variance are respectively expressed as

$$\begin{aligned} E[X(t_{j+1})-X(t_j)|X(t_j)=x] &= \mu(x,t_j)\Delta t, \\ \mathrm{Var}[X(t_{j+1})-X(t_j)|X(t_j)=x] &= \sigma^2(x,t_j)\Delta t. \end{aligned}$$

The corresponding moments on the trinomial tree model are

$$\begin{aligned} &E[X(t_{j+1})-X(t_j)|X(t_j)=x_i]=(p_u(x_i,t_j)-p_d(x_i,t_j))\Delta x, \\ &\mathrm{Var}[X(t_{j+1})-X(t_j)|X(t_j)=x_i]=(p_u(x_i,t_j)+p_d(x_j,t_j))\Delta x^2. \end{aligned}$$

Equating these moments and considering the normalization condition $p_u(x,t)+p_m(x,t)+p_d(x,t)=1$, we obtain

$$p_u(x_i,t_j) = \frac{1}{2}\left(\frac{\sigma^2(x_i,t_j)}{\Delta x^2}+\frac{\mu(x_i,t_j)}{\Delta x}\right)\Delta t, \tag{3.3}$$

$$p_d(x_i,t_j) = \frac{1}{2}\left(\frac{\sigma^2(x_i,t_j)}{\Delta x^2}-\frac{\mu(x_i,t_j)}{\Delta x}\right)\Delta t, \tag{3.4}$$

$$p_m(x_i,t_j) = 1-\frac{\sigma^2(x_i,t_j)}{\Delta x^2}\Delta t. \tag{3.5}$$

In summary, the trinomial tree-model approximates the original SDE by discretizing it on the lattice and setting the transition probabilities between the nodes to reproduce the first and the second moments of the process.

The trinomial tree model simulates the SDE as follows. First, the closest value to x_{ini} in $\{x_i\}_{i\in[0,N_x]}$ is set to x_{i_0}, and the probabilities are set as $\mathrm{Prob}[X(t_0)=x_{i_0}]=1, \mathrm{Prob}[X(t_0)=x_{i\neq i_0}]=0$. Next, the probability distribution of $X(t_1=\Delta t)$ is calculated using the transition probabilities given by Eqs. (3.3)(3.4)(3.5). Repeating this step for $X(t_j)(j=2,3,...,N_t-$

1) yields all probabilities $\mathrm{Prob}[X(t_j) = x_i]$ at node (i, j), from which any properties related to the process $X(t)$, such as the expectation values of $X(T)$ under some function f, $E[f(X(T))]$, can be determined. In option-pricing financial problems, the nodes of the tree model denote the prices of the option, and the problems are sometimes to be solved in the backward direction from time t. In such cases, the boundary condition is set at $t = T$.

3.3 Solving stochastic differential equations by variational quantum simulation

This section presents one of our main results. The SDE simulated by the above-described trinomial tree model is reformulated as the non-unitary dynamics of a quantum state $\left|\tilde{\psi}(t)\right\rangle$ embedding the probability distribution of the random variable $X(t)$. We explicitly state for the $L(t)$ operator of the VQS and decompose it by the polynomial number of the sum of easily-implementable unitaries.

3.3.1 Embedding the probability distribution into a quantum state

To simulate the trinomial tree model of the target SDE by VQS, we define an unnormalized quantum state containing the discretized probability distribution of the random variable $X(t_j)$:

$$\left|\tilde{\psi}(t)\right\rangle \equiv \sum_{i=0}^{N_x} \mathrm{Prob}[X(t) = x_i] \left|i\right\rangle , \tag{3.6}$$

where $\{|i\rangle\}_{i=0}^{N_x}$ is the computational basis. We call this state a directly embedded state. For simplicity, we assume that $N_x = 2^n - 1$, where n is the number of qubits.

Note that this embedding of the probability distribution into the quantum state differs from most of the literature, in which (aiming for a quantum advantage) the expectation values of a probability distribution are calculated using QAE [6]. In the literature, the probability distribution is expressed as a normalized quantum state

$$\left|\psi_{\mathrm{sqrt}}\right\rangle \equiv \sum_i \sqrt{\mathrm{Prob}[X(t_j) = x_i]} \left|i\right\rangle . \tag{3.7}$$

The expectation value of the distribution, $E[f(X(t_j))] \equiv \sum_i f(x_i)\mathrm{Prob}[X(t_j) = x_i]$ for some function f, is computed by the QAE. In this embedding method, VQS cannot be used because the differential equation describing the time evolution of the quantum state is nonlinear. There are ways to solve the

nonlinear differential equation with a quantum algorithm [56, 17, 12], but they require more complicated quantum circuits.

Because our embedding (3.6) differs from this embedding scheme, we also developed a method for evaluating its expectation values (see Sec. 3.4).

3.3.2 Reformulating the trinomial tree model and applying the variational quantum simulation

In the trinomial tree model, the probability $\mathrm{Prob}[X(t_{j+1}) = x_i]$ is calculated as

$$\begin{aligned}\mathrm{Prob}[X(t_{j+1}) = x_i] &= p_u(x_{i-1}, t_j)\mathrm{Prob}[X(t_j) = x_{i-1}] \\ &+ p_d(x_{i+1}, t_j)\mathrm{Prob}[X(t_j) = x_{i+1}] \\ &+ p_m(x_i, t_j)\mathrm{Prob}[X(t_j) = x_i]. \end{aligned} \tag{3.8}$$

Substituting the transition probabilities (3.3), (3.4) and (3.5) into this expression and denoting $P(x,t) \equiv \mathrm{Prob}[X(t) = x]$, we get

$$\begin{aligned}&\frac{P(x_i, t_{j+1}) - P(x_i, t_j)}{\Delta t} \\ =\ & \frac{1}{2}\left(\frac{\sigma^2(x_{i-1}, t_j)}{\Delta x^2} + \frac{\mu(x_{i-1}, t_j)}{\Delta x}\right) P(x_{i-1}, t_j) \\ +\ & \frac{1}{2}\left(\frac{\sigma^2(x_{i+1}, t_j)}{\Delta x^2} - \frac{\mu(x_{i+1}, t_j)}{\Delta x}\right) P(x_{i+1}, t_j) \\ -\ & \frac{\sigma^2(x_i, t_j)}{\Delta x^2} P(x_i, t_j). \end{aligned} \tag{3.9}$$

In the limit $\Delta t \to 0$, one obtains

$$\frac{d\boldsymbol{P}(t)}{dt} = L(t)\boldsymbol{P}(t), \tag{3.10}$$

$$(L(t))_{ik} = \begin{cases} \frac{1}{2}\left(\frac{\sigma^2(x_k,t)}{\Delta x^2} + \frac{\mu(x_k,t)}{\Delta x}\right) & (i = k+1) \\ \frac{1}{2}\left(\frac{\sigma^2(x_k,t)}{\Delta x^2} - \frac{\mu(x_k,t)}{\Delta x}\right) & (i = k-1) \\ -\frac{\sigma^2(x_k,t)}{\Delta x^2} & (i = k) \\ 0 & \text{otherwise} \end{cases}, \tag{3.11}$$

where $\boldsymbol{P}(t) \equiv (P(x_0,t), P(x_1,t), \ldots, P(x_{2^n-1},t))^T$.

As shown in Eq. (3.10), the time evolution of the state $\left|\tilde{\psi}(t)\right\rangle$, or

$$\frac{d}{dt}\left|\tilde{\psi}(t)\right\rangle = \hat{L}(t)\left|\tilde{\psi}(t)\right\rangle, \tag{3.12}$$

where

$$\hat{L}(t) \equiv \sum_{i,k=0}^{2^n-1} (L(t))_{ik} \, |i\rangle\langle k| \,, \tag{3.13}$$

corresponds to the time evolution of the probability distribution $\{\mathrm{Prob}[X(t) = x_i]\}_{i=0}^{2^n-1}$. Equation (3.12) is the essence of our proposal to simulate VQS-based SDE simulation: specifically, the VQS algorithm applied to Eq. (3.12) obtains the time-evolved probability distribution as the quantum state $\left|\tilde{\psi}(t)\right\rangle$. Hereafter, when the distinction is clear in context, we denote the operator $\hat{L}(t)$ by $L(t)$ as in Eq. (3.10).

3.3.3 Construction of $L(t)$

As explained in the previous section, in the VQS, we evaluate Eqs. (2.55) and (2.56), and decomposes $L(t)$ into a sum of easily-implementable unitaries (composed of single-qubit, two-qubit, and few-qubit gates). These evaluations are important for a feasible VQS. This subsection discusses the explicit decomposition of $L(t)$ given by Eq. (3.13).

To express the operator $L(t)$ in Eq. (3.13), we define operators

$$V_+(n) \equiv \sum_{i=0}^{2^n-2} |i+1\rangle\langle i| \,, V_-(n) \equiv \sum_{i=1}^{2^n-1} |i-1\rangle\langle i| \,. \tag{3.14}$$

These operators can be constructed from the n-qubit cyclic increment/decrement operator

$$\mathrm{CycInc}(n) \equiv \sum_{i=0}^{2^n-1} |i+1\rangle\langle i| \,, \; \mathrm{CycDec}(n) \equiv \sum_{i=0}^{2^n-1} |i-1\rangle\langle i| \,, \tag{3.15}$$

where $|-1\rangle, |2^n\rangle$ are identified with $|2^n-1\rangle, |0\rangle$, respectively. These gates are implemented as a product of $O(n)$ Toffoli, CNOT, and X gates with $O(n)$ ancilla qubits [51]. $V_+(n)$(resp. $V_-(n)$) is constructed from CycInc(n)(resp. CycDec(n)) and an n-qubit-control Z gate $C^nZ \equiv \sum_{i=0}^{2^n-2} |i\rangle\langle i| - |2^n-1\rangle\langle 2^n-1|$, which can be implemented [41] as a product of $O(n^2)$ Toffoli, CNOT, and single qubit gates. Using $\frac{1}{2}\left(C^nZ + I^{\otimes n}\right) = \sum_{i=0}^{2^n-2} |i\rangle\langle i|$, we can show that

$$V_+(n) = \mathrm{CycInc}(n) \cdot \frac{1}{2}\left(C^nZ + I^{\otimes n}\right), \tag{3.16}$$

$$V_-(n) = \frac{1}{2}\left(C^nZ + I^{\otimes n}\right) \cdot \mathrm{CycDec}(n), \tag{3.17}$$

meaning that $V_\pm(n)$ can be decomposed into a sum of two unitaries com-

posed of $O(n^2)$ few-qubit gates. Finally, we define the operator $D(n)$ by

$$\begin{aligned} D(n) &= \sum_{i=0}^{2^n-1} i\,|i\rangle\langle i| \\ &= \frac{2^n-1}{2} I^{\otimes n} - \sum_{i=1}^{n} 2^{n-i-1} Z_i, \end{aligned} \tag{3.18}$$

where Z_i is a Z gate acting on the ith qubit. Therefore, $D(n)$ is a sum of $O(n)$ unitaries composed of a single-qubit gate. It follows that

$$V_+(n)(D(n))^m = \sum_{i=0}^{2^n-2} i^m\,|i+1\rangle\langle i|\,, \tag{3.19}$$

$$V_-(n)(D(n))^m = \sum_{i=1}^{2^n-1} i^m\,|i-1\rangle\langle i|\,. \tag{3.20}$$

Let us recall

$$\begin{aligned} L(t) &= \sum_{i=0}^{2^n-2} \frac{1}{2}\left(\frac{\sigma^2(x_i,t)}{\Delta x^2} + \frac{\mu(x_i,t)}{\Delta x}\right)|i+1\rangle\langle i| \\ &+ \sum_{i=1}^{2^n-1} \frac{1}{2}\left(\frac{\sigma^2(x_i,t)}{\Delta x^2} - \frac{\mu(x_i,t)}{\Delta x}\right)|i-1\rangle\langle i| \\ &- \sum_{i=0}^{2^n-1} \frac{\sigma^2(x_i,t)}{\Delta x^2}\,|i\rangle\langle i|\,. \end{aligned}$$

Expanding $\sigma^2(x_i,t)$ and $\mu(x_i,t)$ as

$$\sigma^2(x_i,t) = \sum_{m=0}^{m_\sigma} a_{\sigma,m}(t) x_i^m, \mu(x_i,t) = \sum_{m=0}^{m_\mu} a_{\mu,m}(t) x_i^m, \tag{3.21}$$

we can decompose $L(t)$ as follows:

$$\begin{aligned} & L(t) \\ = & \sum_{m=0}^{m_\sigma} a_{\sigma,m}(t)(\Delta x)^{m-2}\left(\frac{V_+(n)+V_-(n)}{2} - I\right)(D(n))^m \\ + & \sum_{m=0}^{m_\mu} a_{\mu,m}(t)(\Delta x)^{m-1}\left(\frac{V_+(n)-V_-(n)}{2} - I\right)(D(n))^m. \end{aligned}$$

$V_+(n)(D(n))^m$, $V_-(n)(D(n))^m$ and $(D(n))^m$ are composed of the sum of $O(n^m)$ unitaries, each composed of $O(n^2)$ few-qubit gates. In typical SDEs, the orders m_σ, m_μ can be set to small values. For example, geometric Brownian motion case, $m = 1$ (see Sec. 3.6). Therefore, the $L(t)$ decomposition realizes a feasible VQE (Eq. (3.12)).

3.4 Calculation of Expectation Values

In the previous section, we propose a method to simulate the SDE by calculating the dynamics of the probability distribution of a random variable $X(t)$ using VQS. However, in many cases, the goal of the SDE simulation is not the probability distribution of $X(t)$, but the expectation value $E[f(X(t))]$ of $X(t)$ for some function f. In this section, we introduce a means of calculating this expectation value.

3.4.1 Problem Setting

Given a function $f(x) : \mathbb{R} \to \mathbb{R}$, we try to calculate the expectation value $E[f(X(T))]$ of the SDE (3.1) at time $t = T$. The expectation value can be explicitly written as

$$E[f(X(T))] \equiv \sum_{i=0}^{2^n-1} f(x_i)\mathrm{Prob}[X(T) = x_i]. \tag{3.22}$$

Here, we assume that $f(x)$ in the interval $[a_k, a_{k+1}] \in \{[0, a_1], \ldots, [a_{d-1}, x_{\max}]\}$, $(k = 0, \ldots, d-1)$ is well approximated by Lth order polynomials $f_k(x) = \sum_{m=0}^{L} a_m^{(k)} x^m$. The additional error from this piecewise polynomial approximation is evaluated in Appendix A1.5. As x is finite, the range of f is also finite. Thus, by shifting the function f by a constant, we can ensure that the range of f is positive and that the expectation value is also positive, i.e. $E[f(X(T))] \geq 0$. In most situations (such as pricing of European call options as we see in Sec. 3.4.3) the number of intervals d does not scale with the number of qubits n.

3.4.2 General formula for calculating expectation values

We now on compute the expectation value (3.22) using the quantum state $\left|\tilde{\psi}(t)\right\rangle$ (Eq. (3.6)). First, we consider a non-unitary operator satisfying

$$S_f \left|0\right\rangle = \sum_{i=0}^{2^n-1} f(x_i) \left|i\right\rangle \tag{3.23}$$

and decompose S_f into a sum of easily-implementable unitaries as $S_f = \sum_i \xi_i Q_i$ with complex coefficients ξ_i. It follows that

$$\left\langle\tilde{\psi}(t)\right| \left(S_f \left|0\right\rangle\!\left\langle 0\right| S_f^\dagger\right) \left|\tilde{\psi}(t)\right\rangle = \left(E[f(X(T))]\right)^2. \tag{3.24}$$

As $\left|0\right\rangle\!\left\langle 0\right| = \frac{1}{2}\left(I - C^n Z \cdot Z^{\otimes n}\right)$ is also a sum of easily-implementable unitaries as explained in the previous subsection, the Hermitian observable

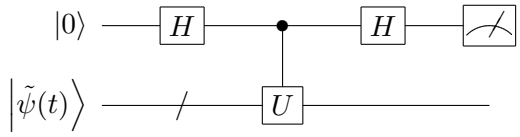


Figure 3.2: Quantum circuit for evaluating the real part of an expectation value Re $\left\langle\tilde{\psi}(t)\middle|U\middle|\tilde{\psi}(t)\right\rangle$ of a unitary operator $U = Q_i Q_{i'}^\dagger, Q_i C^n Z \cdot X^{\otimes n} Q_{i'}^\dagger$. The imaginary part of the expectation value Im $\left\langle\tilde{\psi}(t)\middle|U\middle|\tilde{\psi}(t)\right\rangle$ is evaluated by the circuit with an $S^\dagger$ gate inserted to the left of the second H gate.

$S_f |0\rangle\langle 0| S_f^\dagger$ is decomposed as

$$S_f |0\rangle\langle 0| S_f^\dagger = \sum_{i,i'} \xi_i \xi_{i'}^* \left(Q_i Q_{i'}^\dagger - Q_i (C^n Z \cdot Z^{\otimes n}) Q_{i'}^\dagger \right), \tag{3.25}$$

which is again a sum of unitaries. With this decomposition, the left-hand side of Eq. (3.24) is computed by evaluating

$$\left\langle\tilde{\psi}(t)\middle|Q_i Q_{i'}^\dagger\middle|\tilde{\psi}(t)\right\rangle, \tag{3.26}$$

$$\left\langle\tilde{\psi}(t)\middle|Q_i (C^n Z \cdot Z^{\otimes n}) Q_{i'}^\dagger\middle|\tilde{\psi}(t)\right\rangle. \tag{3.27}$$

Because we set $E[f(X(T))] \geq 0$, the left hand side of Eq. (3.24) will determine the expectation value.

There are two options to evaluate the quantities $\left\langle\tilde{\psi}(t)\middle|Q_i Q_{i'}^\dagger\middle|\tilde{\psi}(t)\right\rangle$ and $\left\langle\tilde{\psi}(t)\middle|Q_i (C^n Z \cdot Z^{\otimes n}) Q_{i'}^\dagger\middle|\tilde{\psi}(t)\right\rangle$. The first one is to use the Hadamard test depicted in Fig. 3.2. The second one is to use quantum phase estimation [57, 58]. The former one requires shallower quantum circuits but is inefficient in terms of the number of measurements to determine the quantities with fixed precision. The detailed computational complexity of these methods is given in Sec. 3.5 and Appendix A1.1.

Next, we explain the construction of the operator S_f in Eq. (3.23) and its decomposition. We first define an operator

$$S_{\chi_{[0,a]}} |0\rangle = \sum_{i=0}^{2^n-1} \chi_{[0,a]}(x_i) |i\rangle = \sum_{x_i \in [0,a]} |i\rangle, \tag{3.28}$$

where $\chi_{[0,a]}(x)$ is the indicator function valued as 1 for $x \in [0,a]$ and 0 for otherwise. Using the binary expansion of $a/\Delta x$, we can obtain the decomposition of $S_{\chi_{[0,a]}}$ hence the decomposition of S_f. As $a \in [0, x_{\max}]$, there exists $k_a \in \mathbb{N}$ such that $\Delta x 2^{k_a - 1} \leq a < \Delta x 2^{k_a}, 0 < k_a \leq n$. The binary expansion of $a/\Delta x$ is given by $a/\Delta x = \sum_{j=0}^{k_a-1} s_j 2^j, s_j \in \{0,1\}$. We

define the list of l as $l_1, l_2, \ldots, l_B (= k_a - 1)$ satisfying $s_l = 1$ in ascending order, and also define an interval

$$\chi_l^a = \left[2^{l_B} + \sum_{j=0}^{l-1} s_j 2^j + 1, 2^{l_B} + \sum_{j=0}^{l} s_j 2^j \right] \tag{3.29}$$

for $l \in \{l_1, l_2, \ldots, l_B\}$. Using χ_l^a, we devide $[0, a/\Delta x]$ into disjoint intervals as follows:

$$[0, a/\Delta x] = [0, 2^{l_B}] \cup \chi_{l_1}^a \cup \cdots \cup \chi_{l_B}^a. \tag{3.30}$$

The indicator operator $S_{\chi_{[0,a]}}$ is obtained by summing the indicator operators on each interval. In binary expansion, the k_ath and the lth bit of $i \in \chi_l^a$ are 1, and the bit below l is either 0 or 1. Accordingly, X should act on the bit taking 1, and H should act on the bit taking either of $\{0, 1\}$. The indicator operator $S_{\chi_l^a}$ on χ_l^a is defined as follows:

$$\begin{aligned} S_{\chi_l^a} |0\rangle &= |0\rangle^{\otimes k_a - 1} \otimes |1\rangle \bigotimes_{j=0}^{n-k_a-l-1} |s_{n-k_a-j}\rangle \otimes \left(\sum_{j=0}^{l} |j\rangle \right) \\ &= 2^{l/2} I^{\otimes k_a - 1} \otimes X \bigotimes_{j=0}^{n-k_a-l-1} \mathbb{X}_{s_{n-k_a-j}} \otimes H^{\otimes l} |0\rangle , \end{aligned} \tag{3.31}$$

where

$$\mathbb{X}_s = \begin{cases} X & (s = 1) \\ I & (s = 0) \end{cases} . \tag{3.32}$$

In addition, we define

$$S_{\chi_{[0,2^{k_a-1}]}} |0\rangle = 2^{(k_a-1)/2} I^{\otimes n-k_a+1} H^{\otimes k_a - 1} |0\rangle . \tag{3.33}$$

We can construct $S_{\chi_{[0,a]}}$ by summing Eqs. (3.31) and (3.33) on each interval. $S_{\chi_{\alpha_k}}$ on interval $\alpha_k \equiv [a_k, a_{k+1}]$ is

$$S_{\chi_{\alpha_k}} = S_{\chi_{[0,a_{k+1}]}} - S_{\chi_{[0,a_k]}}, \tag{3.34}$$

which is a sum of at most $O(n)$ unitaries composed of $O(n)$ gates. Using $S_{\chi_{\alpha_k}}$, we obtain

$$S_f = \sum_{k=0}^{d-1} \sum_{m=0}^{L} a_m^{(k)} (D(n))^m S_{\chi_{\alpha_k}}. \tag{3.35}$$

In summary, evaluation of the expectation value is calculated by the following steps.

1. Divide the domain of the target function $[0, x_{\max}]$ into intervals $[a_k, a_{k+1}] \in \{[0, a_1], [a_1, a_2], \ldots, [a_{d-1}, x_{\max}]\}$.
2. Approximate the function in each interval $[a_k, a_{k+1}]$ by Eq. (3.35).
3. Decompose $S_f |0\rangle \langle 0| S_f^\dagger$ into a sum of unitary terms and calculate each term using the circuits in Fig. 3.2.

As $S_{\chi_{[a_k,a_{k+1}]}}, (D(n))^m, |0\rangle \langle 0|$ is the sum of $O(n)$, $O(n^m)$ and $O(1)$ unitaries composed of $O(n)$, $O(1)$ and $O(n^2)$ gates, respectively, $S_f |0\rangle \langle 0| S_f^\dagger$ is the sum of $O(d^2 n^{2L+2})$ unitaries and each Q_i is composed of at most $O(n^4)$ gates.

When the target function f on each interval is written by a low-degree polynomial (i.e., L is small), especially by a linear function (as in the pricing of European call options shown below), our algorithm can efficiently calculate the expectation value because the number of unitaries $O(d^2 n^{2L+2})$ gets not so large. When the function f is approximated by the polynomial, we can estimate the error of the expectation value stemming from that approximation. If we want to suppress the error below ϵ, the number of unitaries becomes $O(x_{\max}^2 \epsilon^{-\frac{2}{L+1}} n^{2L+2})$ (the derivation is presented in Appendix A1.5). Note that as L is increased, $\epsilon^{-\frac{2}{L+1}}$ becomes smaller while n^{2L+2} becomes larger. The number of unitaries, therefore, is not monotonic with respect to L, and there may be an optimal L for the desired accuracy. We note that evaluation of expectation values of those unitaries can be performed completely in parallel by independent quantum devices.

3.4.3 Pricing of The European Call Option

As a concrete example, we present the pricing of a European call option with the BS model. The price of a European call option with strike price K, risk-free interest rate $r \geq 0$, and maturity $T \geq 0$ is defined by the conditional expected value

$$e^{-rT} E_Q \left[\max(X_T - K, 0) | X_0 = x_0\right]. \tag{3.36}$$

Stochastic processes are assumed to follow geometric Brownian motion in the BS model, but are described by more complex mechanisms in other models. Even in these models, the expression Eq. (3.36) of the price of the European call option is the same with the present case as in Eq. (2.6).

Setting the probability distribution of X_T conditioned by $X_0 = x_0$ as $\{\text{Prob}\left[X_T = x_i | X_0 = x_0\right]\}_{i=0}^{2^n-1}$, the expectation value is

$$\begin{aligned} & e^{-rT} E_Q \left[\max(X_T - K, 0) | X_0 = x_0\right] \\ = \; & e^{-rT} \sum_{i=0}^{2^n-1} \text{Prob}\left[X_T = x_i | X_0 = x_0\right] \max(x_i - K, 0). \end{aligned} \tag{3.37}$$

For simplicity, we assume $\Delta x = 1$ and $K = 2^k < 2^n - 1, k \in \mathbb{N}$. We thus obtain

$$S_{\max(i-K,0)} = (D(n) - KI)S_{\chi_{[K,2^n-1]}}. \tag{3.38}$$

In this case, there are only two intervals $[0, K-1]$ and $[K, 2^{n-1}]$, and the polynomial in each interval is of first-order degree at most. Therefore, we can calculate the price of the European call option by Eq. (3.24).

3.5 Possible Advantages of Our Method

In this section, we discuss the advantages of our method compared to previous studies, as well as the possible quantum advantages.

In general, the SDEs addressed in this chapter can be transformed into a PDE of the function $e_f(x,t)$, where $e_f(x,t)$ gives the expectation value $E[f(X(T-t))|X(0) = x]$, by Feynman-Kac formula as described in Sec. 2.1.2. In fact, the authors of [29] performed a variational quantum computation of a PDE of this function. We point out two advantages of our method compared with this strategy using Feynman-Kac formula. First, the resulting PDE by Feynman-Kac formula must be solved backwardly in time from $t = T$ to $t = 0$, with the initial condition at $t = T$ being related to the functional form of $f(X)$. It is not trivial to prepare the initial state $|\psi(T)\rangle$ corresponding to the initial condition; the authors of [48] executed an additional VQE to prepare the initial state. Second, when using the Feynman-Kac formula, the initial condition of the PDE is different for each function f for which we want to calculate the expectation value $E[f(X(T))]$. If we want to calculate a different expectation value $E[f'(X(T))]$, we need to run the whole algorithm simulating the PDE with the different initial state corresponding to f'. On the other hand, in our method, once we perform VQS, we obtain the probability distribution of $X(T)$ as a quantum state and the corresponding variational parameters to reproduce it. We only need to redo the part of the expectation value calculation (Sec. 3.4) for different f'.

The authors of [19] embedded the probability distribution by quantum arithmetic. Their embedding, proposed in [11], requires $O(2^n)$ gates to embed the probability distribution into an n-qubit quantum state. To moderate the gate complexity, the authors of [15] embedded the probability distribution using a quantum generative adversarial network, which requires only $O(\mathrm{Poly}(n))$ gates. The probability distribution function can also be approximated by a lth-order piecewise polynomial, which can be embedded with $O(ln^2)$ gates even in quantum arithmetic [59]. However, both methods require prior knowledge of the probability distribution to be embedded. In contrast, our method does not require prior knowledge of the embedding probability distribution since our method simulates the time evolution of a given SDE.

We now compare the computational cost to calculate expectation values with previous studies. In [19] and [15], by employing QAE, the expectation value (Eq. (3.22)) was calculated by using an oracle that is complex quantum gate reflecting the functional form of f for $O(1/\epsilon)$ times, where ϵ is the precision for the expectation values. The classical Monte Carlo method requires $O(1/\epsilon^2)$ sampling for precision ϵ, so their methods provide a second-order acceleration. On the other hand, our method measures the expectation value of each term of Eq. (3.25) using the Hadamard test (Fig. 3.2) or the quantum phase estimation (QPE) [57, 58]. As shown in Appendix A1.1, the total number of measurements to obtain the expectation value with precision ϵ is $O(1/\gamma\epsilon^2)$ for the Hadamard test and $O(\log(1/\gamma\epsilon))$ for QPE, where γ is some factor. We note that the depth of the circuit is $O(1/\gamma\epsilon)$ in QPE, which is in the same order as the QAE whereas our method requires not an complicated oracle but a relatively-small unitary. Hence, when the factor γ is not too small, our method combined with QPE can also exhibit quantum advantage for the evaluation of the expectation values. The factor γ depends on the parameters of the polynomial approximation $(a_k^{(m)}, d, L)$, the domain of the approximated function $x_{\max}$, and the probability distribution $\{\mathrm{Prob}[X = x_i]\}_{i=0}^{2^n-1}$. The detailed evaluation of γ is described in Appendix A1.1.

3.6 Numerical Results

In this section, our algorithm is applied to two stochastic processes, namely, geometric Brownian motion and an Ornstein-Uhlenbeck process, which are commonly assumed in financial engineering problems. Geometric Brownian motion simply models the fluctuations of asset prices, and the Ornstein-Uhlenbeck process is a popular model of interest rates.

3.6.1 Models

Geometric Brownian motion is equivalent to setting $\mu(X(t), t) = rX(t)$, $\sigma(X(t), t) = \sigma X(t)$ in Eq. (3.1), where r and σ are positive constants.

The Ornstein–Uhlenbeck process is equivalent to setting $\mu(X(t), t) = -\eta(X(t) - r), \sigma(X(t), t) = \sigma$ in Eq. (3.1), where η, r and $\sigma > 0$ are constants.

The ansatz circuit is identical for both models and shown in Fig. 3.3. As the amplitudes of the quantum state must be real, the ansatz contains only CNOT and RY gates. This depth-k circuit repeats the entangle blocks composed of CNOTs and RY gates k times. The parameters of geometric Brownian motion were $r = 0.1, \sigma = 0.2, \Delta x = 1$, and $t \in [0, 4]$ and those of the Ornstein–Uhlenbeck process were $r = 7, \sigma = 0.5, \eta = 0.01, \Delta x = 1$, and $t \in [0, 4]$. We simulate the quantum circuits without noise using numpy [60] and jax [61]. We set the number of qubits $n = 4$ and the number of repetitions of entangle blocks $k = 2, 3$.

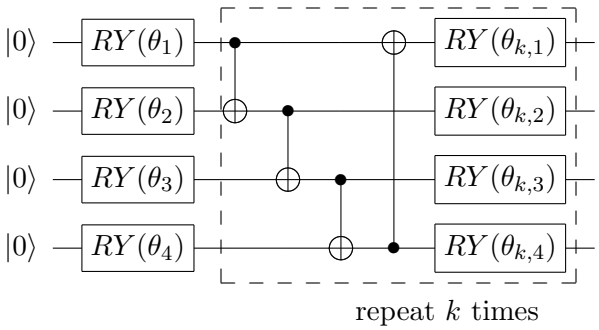
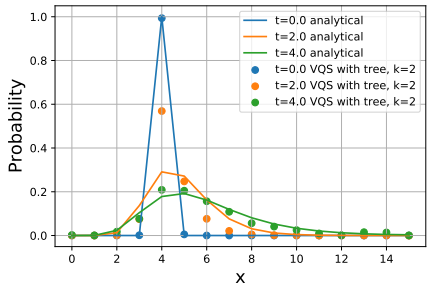


Figure 3.3: In a depth-k circuit, CNOT and RY gates (enclosed by dashed lines) are repeated k-times. The circuit has $4(k+1)$ parameters.

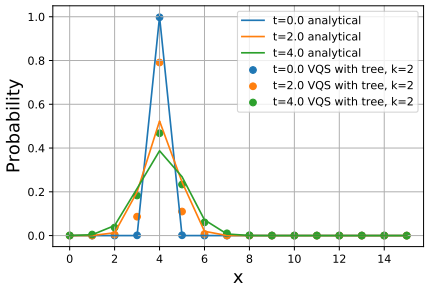
3.6.2 Results

Panels (a) and (b) of Fig. 3.4 present the numerical simulations of geometric Brownian motion and the Ornstein-Uhlenbeck process, respectively. In comparison, we also provide a probability density function (PDF) for the solution of the SDE equation obtained by solving the Fokker-Planck equation [62] analytically. We can see that our method describes the time evolution of the probability distribution well. Note that in the range where t is small, the probability is locally distributed and the width of the difference is large relative to the spread of the distribution. Therefore, the error due to the finite difference method is also large. This error may be suppressed by varying the width of the difference in time and space, instead of limiting the grid to equally spaced intervals. We leave the suppression of errors when t is small as a future work.

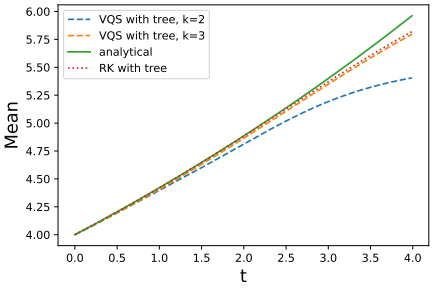
We calculated the means (Fig. 3.4(c),(d)) and variances (Fig. 3.4(e),(f)) of the resulting distributions. We also present the mean and variance obtained from the analytical solution and the solution of (3.10) using the Runge-Kutta method. Because of the approximation with the tree model, even the results of the Runge-Kutta method slightly differ from the analytical solution. In the case of VQS with $k = 2$, we see that the error from the analytical solution is larger than that of the $k = 3$ case. This is because the number of VQS parameters is less than the number of lattice points in the event space when $k = 2$, i.e., the degrees of freedom of ansatz are less than the degrees of freedom of the system, and thus the errors due to the ansatz appear. In the case of $k = 3$, the number of parameters in ansatz is sufficient, and thus the results are closer to the results of the Runge-Kutta method.



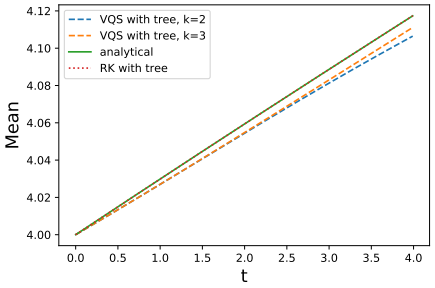
(a) Dynamics of geometric Brownian motion



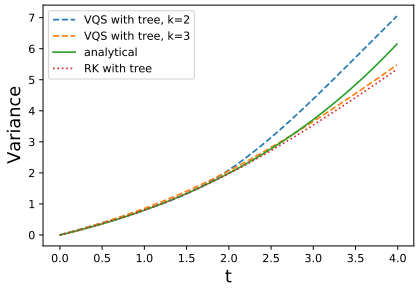
(b) Dynamics of Ornstein-Uhlenbeck process



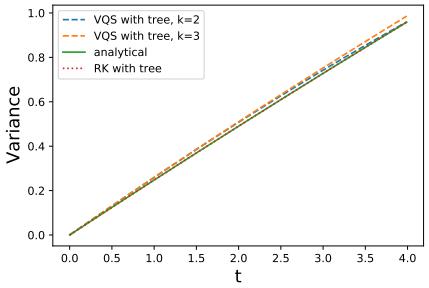
(c) Time dependence of mean in geometric Brownian motion



(d) Time dependence of mean in Ornstein-Uhlenbeck process



(e) Time dependence of variance in geometric Brownian motion



(f) Time dependence of variance in Ornstein-Uhlenbeck process

Figure 3.4: (a), (b): Exact solutions of the SDE (solid lines) and numerical simulation of our algorithm (circles); (c), (d): time dependence of the means and (e), (f) variances of the encoded probability distributions. Dashed lines show the numerical solutions of VQS with a $k = 2, 3$ depth ansatz. Dotted lines show the numerical Runge-Kutta solutions of the linear differential equation (Eq. (3.10)). Solid lines show the exact solutions of the SDE. (a), (c), (e): Geometric Brownian motion with parameters $r = 0.1, \sigma = 0.2$. (b), (d), (f): Ornstein-Uhlenbeck Process with parameters $r = 7, \sigma = 0.5, \eta = 0.01$.

3.7 Conclusion

This chapter proposed a quantum-classical hybrid algorithm that simulates SDEs based on VQS. A continuous stochastic process was discretized in a trinomial tree model and was reformulated as a linear differential equation. The obtained differential equation was solved with VQS, obtaining quantum states representing the probability distribution of the stochastic processes. As our method can embed the probability distribution of the solution of a given SDE into the quantum state, it is applicable to general SDEs. We note that our methods can apply to the Fokker-Plank equation, which also gives the time-evolution of the probability distributions of SDE solutions.

Because the embedding methods of the probability distribution differ in the proposed method and the conventional quantum algorithm, we proposed another method for computing the expectation value. We approximated the functions to calculate expectation values by piecewise polynomials and constructed operators corresponding to the polynomial in each interval. The operators were constructed as sums of unitary operators, which are composed of easily-implementable gates. The expectation value was then computed using the sum of unitary operators. Our algorithm was validated in classical simulations of geometric Brownian motion and the Ornstein-Uhlenbeck process. Both processes were well simulated by the algorithm. Our algorithm is expected to efficiently simulate other stochastic processes, provided that $L(t)$ can be written as a polynomial linear combination of unitary matrices.

Let us summarize the computational cost of our method presented in this work. Our method consists of two parts; one is to perform VQS to simulate the SDE, and the other is to calculate the expectation value of the SDE solution. In the part of running VQS, we decompose matrix $L(t)$ in Eq. (3.10) into a sum of $O(n^{m_{\max}})$ different unitaries composed of $O(n^2)$ few-qubit gates, where $m_{\max}$ is the largest order of the polynomial expansion of μ, σ in Eq. (3.21). At each time step of VQS, the vector V_k in Eq. (2.56) is evaluated as a sum of $O(n^{m_{\max}})$ measurement results of the circuits depicted in Fig. 2. As $m_{\max}$ is typically finite and small ($\sim 1, 2$) in most practical applications, the computational cost (i.e., the number of gates in quantum circuits, the number of different circuits to run) of the simulation of SDE is $O(\mathrm{Poly}(n))$. In contrast, QLSA [20, 54, 55, 4] requires much deeper and more complex quantum circuits and a large number of ancilla qubits because it uses the Hamiltonian simulation and the quantum Fourier transform. This is an advantage of our method leveraging the variational quantum algorithm.

In the part of the expectation value evaluation of the SDE solution, we evaluate it by running different $O(d^2 n^{2L+2})$ quantum circuits, where d and L are the number of intervals and the order of the piecewise polynomial approximation of the function f in Eq. (3.22), respectively. Each circuit is constructed to compute an expectation value $\langle\psi|U|\psi\rangle$ of a unitary U that contains $O(n^4)$ quantum gates. When we adopt the Hadamard test (Fig. 3.2)

as such a quantum circuit, the number of measurements to suppress statistical error of the expectation value below ϵ is $O(1/\gamma\epsilon^2)$, where γ is a factor defined in Appendix A1.1. This $O(1/\epsilon^2)$ scaling is the same as the classical Monte Carlo method to compute the expectation values from the probability distribution of the SDE solution. When we choose the QPE-type circuit to evaluate $\langle\psi|U|\psi\rangle$, the number of measurements becomes $O(\log(1/\gamma\epsilon))$ while the depth of the circuit in terms of U is $O(1/\gamma\epsilon)$. This situation can provide a quantum advantage for computing the expectation value of the SDE solution. The error from the piecewise polynomial approximation of f can be made small by increasing d or L, which is detailed in Appendix A1.5. Since it is difficult to accurately estimate the error caused by ansatz, it is also difficult to accurately estimate the overall error. Therefore, quantum speedup is not mathematically rigorous as with other variational quantum algorithms.

This study focused on computational finance because financial engineering is among the most popular applications of stochastic processes. Pricing of derivatives, and many other problems in financial engineering, satisfy the conditions of the proposed method. However, as the stochastic processes themselves are quite general, the proposed method is expected to contribute to solving problems in various fields.

Chapter 4

Pricing multi-asset derivatives by finite difference method on a quantum computer

In this chapter, we present the quantum algorithm that calculate the price of the multi-asset derivative with finite difference method on a fault-tolerant quantum computer. As described in Sec. 2.2.2, the finite difference method requires $O(n_{\mathrm{gr}}^d)$ grid points for pricing d-asset derivatives. Several previous studies [27, 28, 29, 30], use quantum computer to simulate the time evolution of the derivative price with the finite difference method to avoid this exponential growth of the required resource. However, the resulting outcomes of these quantum algorithms are *quantum information*, i.e., the quantum states where the derivative prices are embedded. The present price of the derivative is one element of the quantum state vector. Thus, to obtain the derivative price as a *classical information*, we need to estimate the element, but it would take exponential number of measurement. As a result, the quantum speedup would be lost. To overcome this issue, we utilize the martingale of the derivative price, i.e. the fact that the present value of the derivative can be calculated by the expected value at arbitrary future time. This chapter is based on [K. Miyamoto, K. Kubo, IEEE Transactions on Quantum Engineering 3, 3100225 (2021) © 2021 IEEE] with modifications for fitting in the context. The content of Secs. II and III of [K. Miyamoto, K. Kubo, IEEE Transactions on Quantum Engineering 3, 3100225 (2021) © 2021 IEEE] is in Secs. 2.2.2 and 2.3.2 respectively, and this chapter depends on the sections.

4.1 Introduction

Recently, people are witnessing the great advance of *quantum computing*[1], which can speedup some computational tasks compared with exiting *classical* computers, and taking a strong interest in its industrial applications. Finance is one of promising fields. Since large financial institutions perform enormous computational tasks in their daily business[2], it is naturally expected that quantum computers will tremendously speedup them and make a large impact on the industry. In fact, some recent papers have already discussed applications of *quantum algorithms* to concrete problems in financial engineering: for example, derivative pricing[19, 63, 64, 65, 29, 66, 31, 67, 68, 69, 27, 28, 30, 70], risk measurement[71, 72, 73], portfolio optimization[74, 75, 76], and so on. See [77, 78, 79] as comprehensive reviews.

Among a variety of applications, we here consider the quantum method for derivative pricing. Especially, we focus on the approach based on solving the partial differential equation (PDE) by finite difference method (FDM). Let us describe the outline of the problem. First of all, we explain what *financial derivatives*, or simply *derivatives* are. They are products whose values are determined by prices of other simple assets (*underlying assets*) such as stocks, bonds, foreign currencies, commodities, and so on. They can be characterized by payoffs paid and/or received between parties involved in a derivative contract, whose amounts are determined by underlying asset prices. One of the simplest examples of derivatives is an European call (resp. put) option, the right to buy (resp. sell) some asset at the predetermined price (*strike*) K and time (*maturity*) T. This is equivalent to the contract that the option buyer receives the payoff $f_{\text{pay}}(S_T) = \max\{S_T - K, 0\}$ (call) or $\max\{K - S_T, 0\}$ (put) at T, where S_t is the underlying asset price at time t. Besides this, there are many types of derivatives, some of which contain complicated contract terms and are called *exotic* derivatives.

Since large banks hold a large number of exotic derivatives, pricing them is crucial for their business. We can evaluate a derivative price by modeling the random time evolution of underlying asset prices by some stochastic processes and calculating the expected values of the payoff[3] under some probability measure. Since analytical formulas for the derivative price are available only in limited settings, we often resort to numerical methods. One of major approaches is Monte Carlo simulation. That is, we generate many paths of underlying asset price evolution on some discretized time grid, and then take the average of payoffs over the paths. We can also take another

[1] For readers who are unfamiliar to quantum computing, we refer to [41] as a standard textbook.

[2] For readers who are unfamiliar to financial engineering or, more specifically, derivative pricing, we refer to [1, 2]

[3] Strictly speaking, the payoff must be divided by some numeraire.

approach: since the expected value obeys some PDE, which is called the Black-Scholes (BS) PDE, we can obtain the derivative price by solving it[4]. More concretely, starting from the maturity T, at which the derivative price is trivially determined as the payoff itself, we solve the PDE backward to the present, and then find the present value of the derivative.

We should choose the appropriate approach according to the nature of the problem. For example, PDE approach is suitable for derivatives whose price is subject to some continuous *boundary conditions*. One prominent example is the *barrier option*. In a barrier option contract, one or multiple levels of underlying asset prices, which are called barriers, are set. Then, they determine whether the payoff is paid at the maturity or not. For example, in a *knock-out* barrier option, the payoff is not paid if either of barriers is reached once or more by T, regardless of S_T[5]. This means that the price of the knock-out barrier option is 0 at barriers. Such a boundary condition is difficult to be strictly taken into account in the Monte Carlo approach because of discretized time evolution, but it can be dealt with in the PDE approach.

Although the PDE approach is suitable in these cases, it is difficult to apply it to *multi-asset derivatives*, that is, the case where the number of underlying assets d is larger than 1. This is because of the exponential growth of complexity with respect to d, which is known as the *curse of dimensionality*. We can see this as follows. The BS PDE is $(d+1)$-dimensional, where d and 1 correspond to asset prices and time, respectively. In FDM, which is often adopted for solving a PDE numerically, the discretization grid points are set in the asset price directions, and partial derivatives are replaced with matrices which correspond to finite difference approximation. This converts a PDE into a linear ordinary differential equation (ODE) system, in which the dependent variables are the derivative prices on grid points and the independent variable is time. Then, we solve the resulting ODE system. The point is that this calculation contains manipulations of the matrices with exponentially large size, that is, $n_{\rm gr}^d \times n_{\rm gr}^d$, where $n_{\rm gr}$ is the number of the grid points in one direction. Since we have to take $n_{\rm gr}$ proportional to $\epsilon^{-1/2}$ in order to accomplish the error level ϵ, as shown later, the time complexity of this approach grows as $O(\mathrm{poly}(\epsilon^{-d/2})) = O((1/\epsilon)^{\mathrm{poly}(d)})$. Besides, the space complexity also grows exponentially, since we have to store the derivative prices on grid points in calculation. This makes the PDE approach, at least in combination with FDM, intractable on classical computers.

[4]For comprehensive reviews of the PDE approach for derivative pricing, we refer to [80, 81] as textbooks.

[5]There are also *knock-in* barrier options, where the payoff is paid only if either of barriers is reached at least once by T. We can price an knock-in barrier option by subtracting the price of the corresponding knock-out barrier option from that of the corresponding European (that is, no-barrier) option, since a combination of a knock-in barrier option and a knock-out barrier option is equivalent to an European option.

Fortunately, quantum computers might change the situation. This is because there are some quantum algorithms for solving linear ODE systems, whose time complexities depend on dimensionality only logarithmically[55, 4, 82, 83]. This means that, in combination with these algorithms, we can remove the exponential dependency of time complexity of FDM on dimensionality. In fact, some quantum algorithms for solving PDE, including not only FDM-based ones but also different approaches, have already been proposed, and quantum speedup is obtained in some cases[45, 22, 84, 85, 86, 87, 88, 89]. Note also that the space complexity can be also reduced exponentially, since, using a n-qubits system, we can encode a vector with exponentially large size with respect to n into the amplitudes of the quantum state.

In light of the above, this chapter aims to speedup FDM-based pricing of multi-asset derivatives, utilizing the quantum algorithm. Although one might think that this is just a straightforward application of an existing algorithm to some problem, there is a nontrivial issue specific for derivative pricing.The issue is how to extract the present value of the derivative from the output of the quantum algorithm. By solving the BS PDE up to the present ($t = 0$) using the quantum algorithm, we obtain the vector $\boldsymbol{V}(0)$, which consists of the derivative prices on the grid points in the space of the underlying asset prices. However, it is given not as classical data but as a quantum state $|\boldsymbol{V}(0)\rangle$, in which the elements of $\boldsymbol{V}(0)$ are encoded as amplitudes of computational basis states. On the other hand, typically, we are interested in only one element of $\boldsymbol{V}(0)$, that is, V_0, the derivative price for the present underlying asset prices. This means that we have to obtain the amplitude of the specific computational basis state in $|\boldsymbol{V}(0)\rangle$. Since the amplitude is exponentially small if the number of the grid point is exponentially large, reading it out requires exponentially large time complexity, which ruins the quantum speedup.

We circumvent this issue by solving PDE up to not the present but some future time t_{ter}. The key observation is that V_0 can be expressed as the expected value of its price[6] at an arbitrary future time. Concretely, we may take the following way. First, we generate two states: $|\boldsymbol{V}(t_{\mathrm{ter}})\rangle$, in which the derivative prices at t_{ter} are encoded, and $|\boldsymbol{p}(t_{\mathrm{ter}})\rangle$, in which the probability distribution of underlying asset prices at t_{ter} are encoded. Then, we estimate the inner product $\langle \boldsymbol{p}(t_{\mathrm{ter}})|\boldsymbol{V}(t_{\mathrm{ter}})\rangle$, which is an approximation of V_0. Note that the amplitude of each basis state in $|\boldsymbol{V}(t_{\mathrm{ter}})\rangle$ contains information to determine V_0 differently from $|\boldsymbol{V}(0)\rangle$, in which the amplitude of one specific basis state is the sole necessary information. This leads to much smaller time complexity in the above way than reading V_0 out from $|\boldsymbol{V}(0)\rangle$.

In the following sections, we describe the entire process of the above calculation: setting t_{ter}, generating $|\boldsymbol{V}(t_{\mathrm{ter}})\rangle$ by the quantum algorithm,

[6] Again, strictly speaking, the price must be divided by some numeraire.

generating $|\boldsymbol{p}(t_{\mathrm{ter}})\rangle$, and estimating V_0. Besides, we estimate the complexity of the proposed method. We see that, in the expression of the complexity, there are not any factors like $(1/\epsilon)^{\mathrm{poly}(d)}$ but only some logarithmic factors to the power of d, which means substantial speedup compared with classical FDM.

The rest of this chapter is organized as follows. Sections 4.2 is preliminary one, which outline the quantum algorithm for solving ODE systems. Here, we also discuss how to set t_{ter} in Section 4.3, taking into account the probability that underlying assets reach the barrier. Section 4.4 presents the main result, that is, the quantum calculation procedure for V_0 and its complexity. Section 4.5 summarizes this chapter. All proofs are shown in the appendix.

4.1.1 Notations

Here, we explain the notations used in this chapter.

- I_n: The $n \times n$ identity matrix.
- $\| \cdot \|$: The Euclidean norm for a vector and the spectral norm for a matrix. We call each of them a "norm" simply.
- $\mu(A)$: The logarithmic norm associated with $\| \cdot \|$ for a $n \times n$ matrix A, that is, $\mu(A) \coloneqq \lim_{h \to 0^+} (\|I_n + hA\| - 1)/h$.
- $|i\rangle$: A state on a multi-qubit register.
- $|\bar{i}\rangle$: A state on one qubit.
- $\| |\psi\rangle \|$: The norm of a (unnormalized) state $|\psi\rangle$.

When a matrix A has at most s nonzero entries in any row and column, we say that the sparsity of A is s. If a state $|\psi\rangle$ satisfies $\| |\psi\rangle - |\psi'\rangle \| < \epsilon$, where ϵ is a positive real number and $|\psi'\rangle$ is another state, we say that $|\psi\rangle$ is ϵ-close to $|\psi'\rangle$.

4.2 Quantum algorithm for solving ODE systems

In this chapter, we consider the Problem 2.2.1. To solve the problem, we use the finite difference method introduced in Section 2.2.2 and the quantum algorithm for solving ODE systems introduced in Section 2.3.2.

The quantum algorithm in [4] leverages the algorithm in [23]. In order to use it to solve Eq. (2.44), [4] assumes that the following oracles (i.e. unitary operators) are available:

- $O_{A,1}$
  For the matrix A, given a row index j and an integer l, this return $\nu(j,l)$, the column index of the l-th nonzero entry in the j-th row:

$$O_{A,1} : |j\rangle |l\rangle \mapsto |j\rangle |\nu(j,l)\rangle \tag{4.1}$$

- $O_{A,2}$
  For the matrix A, given a row index j and a column index k, this return the (j,k) entry:

$$O_{A,2} : |j\rangle |k\rangle |z\rangle \mapsto |j\rangle |k\rangle |z \oplus A_{jk}\rangle \tag{4.2}$$

- $O_{\boldsymbol{x}_{\mathrm{ini}}}$
  This prepares $\frac{1}{\|\boldsymbol{x}_{\mathrm{ini}}\|} |\boldsymbol{x}_{\mathrm{ini}}\rangle$ under the control by another qubit:

$$O_{\boldsymbol{x}_{\mathrm{ini}}} : \begin{cases} |\bar{0}\rangle |0\rangle \mapsto \frac{1}{\|\boldsymbol{x}_{\mathrm{ini}}\|} |\bar{0}\rangle |\boldsymbol{x}_{\mathrm{ini}}\rangle \\ |\bar{1}\rangle |\psi\rangle \mapsto |\bar{1}\rangle |\psi\rangle \quad \text{for any} \quad |\psi\rangle \end{cases} \tag{4.3}$$

- $O_{\boldsymbol{b}}$
  When $\boldsymbol{b} \neq 0$, this prepares $\frac{1}{\|\boldsymbol{b}\|} |\boldsymbol{b}\rangle$ under the control by another qubit:

$$O_{\boldsymbol{b}} : \begin{cases} |\bar{0}\rangle |\psi\rangle \mapsto |\bar{0}\rangle |\psi\rangle \quad \text{for any} \quad |\psi\rangle \\ |\bar{1}\rangle |0\rangle \mapsto \frac{1}{\|\boldsymbol{b}\|} |\bar{1}\rangle |\boldsymbol{b}\rangle \end{cases} . \tag{4.4}$$

  When $\boldsymbol{b} = 0$, this is an identity operator.

Then, we present the theorem (Theorem 9 in [4]), which states the query complexity of the quantum algorithm for solving ODE systems discussed in Sec. 2.3.2, with a slight modification.

Theorem 4.2.1. *(Theorem 9 in [4], slightly modified) Suppose $A = VDV^{-1}$ is an $N \times N$ diagonalizable matrix, where $D = \mathrm{diag}(\lambda_0, \lambda_1, ..., \lambda_{N-1})$ satisfies $\mathrm{Re}(\lambda_j) \leq 0$ for any $j \in 0, 1, ..., N-1$. In addition, suppose A has at most s nonzero entries in any row and column, and we have oracles $O_{A,1}, O_{A,2}$ as above. Suppose $\boldsymbol{x}_{\mathrm{ini}}$ and $\boldsymbol{b}$ are N-dimensional vectors with known norms and we have oracles $O_{\boldsymbol{x}_{\mathrm{ini}}}$ and $O_{\boldsymbol{b}}$ as above. Let $\boldsymbol{x}$ evolve according to the differential equation (2.43) with the initial condition $\boldsymbol{x}(0) = \boldsymbol{x}_{\mathrm{ini}}$. Let $T > 0$ and $g := \max_{t \in [0,T]} \|\boldsymbol{x}(t)\| / \|\boldsymbol{x}(T)\|$. Then there exists a quantum algorithm that produces a state $\left|\tilde{\Psi}\right\rangle$, which is ϵ-close to*

$$|\Psi\rangle := \frac{1}{\sqrt{\langle \Psi_{\mathrm{gar}} | \Psi_{\mathrm{gar}} \rangle + (p+1) \|\boldsymbol{x}(T)\|^2}} \left(|\Psi_{\mathrm{gar}}\rangle + \sum_{j=p(k+1)}^{p(k+2)} |j\rangle |\boldsymbol{x}(T)\rangle \right) \tag{4.5}$$

using

$$O\left(\kappa_V sT\|A\| \times \mathrm{poly}\left(\log\left(\frac{\kappa_V sT\|A\|}{\epsilon}\right)\right)\right) \tag{4.6}$$

queries to $O_{A,1}$, $O_{A,2}$, O_x, *and* O_b. *Here,* $\kappa_V = \|V\| \cdot \|V^{-1}\|$ *is the condition number of* V, $p = \lceil T\|A\|\rceil$, $k = \lfloor 2\log\Omega/\log(\log\Omega)\rfloor$, $\Omega = 70g\kappa_V p^{3/2}(\|\boldsymbol{x}_{\mathrm{ini}}\|+T\|\boldsymbol{b}\|)/\epsilon\|\boldsymbol{x}(T)\|$, *and* $|\Psi_{\mathrm{gar}}\rangle$ *is an unnormalized state which takes the form of* $|\Psi_{\mathrm{gar}}\rangle = \sum_{j=0}^{p(k+1)-1}|j\rangle|\psi_j\rangle$ *with some unnormalized states* $|\psi_0\rangle$, $|\psi_1\rangle$, ..., $|\psi_{p(k+1)-1}\rangle$ *and satisfies* $\langle\Psi_{\mathrm{gar}}|\Psi_{\mathrm{gar}}\rangle = O(g^2(p+1)\|\boldsymbol{x}(T)\|^2)$.

The modifications from Theorem 9 in [4] are as follows. First, in [4], it is assumed that we perform post-selection and obtain $|\boldsymbol{x}(T)\rangle/\||\boldsymbol{x}(T)\rangle\|$ (strictly speaking, a state close to it). On the other hand, in Theorem 4.2.1, the output state is not purely $|\boldsymbol{x}(T)\rangle/\||\boldsymbol{x}(T)\rangle\|$ but contains $|\boldsymbol{x}(T)\rangle$ as a part in addition to the unnecessary state $|\Psi_{\mathrm{gar}}\rangle$. This is because, in this chapter, we use the algorithm of [4] as a subroutine in the quantum amplitude estimation (QAE)[6, 90, 91, 92, 93], as explained in Section 4.4, and the iterated subroutine in QAE must be an unitary operation. This means that we cannot perform post-selection, since it is a non-unitary operation. Note also that, we do not perform amplitude amplification for $|\Psi_1\rangle$, which is done before post-selection in [4], and thus a factor g, which exists in the expression of the complexity (112) in [4], has dropped from (4.6) in this chapter. Moreover, the meaning of the closeness ϵ is different between Theorem 4.2.1 in this chapter and Theorem 9 in [4]. In the former, ϵ is the closeness between $\left|\tilde{\Psi}\right\rangle$ and $|\Psi\rangle$, which corresponds to δ in [4]. On the other hand, Theorem 9 in [4] refers to the closeness of the state after post-selection to $|\boldsymbol{x}(T)\rangle/\||\boldsymbol{x}(T)\rangle\|$. This difference also makes (4.6) different from (112) in [4].

4.3 Approximating the present derivative price as the expected value of the price at a future time

As we explained in the preliminary, we aim to calculate V_0 as the expected value of the discounted price at some future time. Concretely, we set $t_{\mathrm{ter}} \in (0,T)$ and calculate

$$V_0 = e^{-rt_{\mathrm{ter}}}\int_{\mathbb{R}_+^d} d\boldsymbol{S}\phi(t_{\mathrm{ter}},\boldsymbol{S})p_{\mathrm{NB}}(t_{\mathrm{ter}},\boldsymbol{S})V(t_{\mathrm{ter}},\boldsymbol{S}), \tag{4.7}$$

where $\phi(t,\boldsymbol{S})$ is the probability density function of $\boldsymbol{S}(t)$, and $p_{\mathrm{NB}}(t,\boldsymbol{S})$ is the conditional probability that the no event which leads to extinction of the payoff happens by t given $\boldsymbol{S}(t) = \boldsymbol{S}$. Although (4.7) holds for any t_{ter}, for the effective numerical calculation, t_{ter} should be set carefully. Recalling our motivation to evade exponential complexity to read out V_0, which is explained in the beginning of this chapter, we want to set t_{ter} as large

as possible. On the other hand, there are some reasons to set t_{ter} small because of existence of boundaries. First, note that it is difficult to find $p_{\mathrm{NB}}(t_{\mathrm{ter}}, \boldsymbol{S})$ explicitly in the multi-asset case. However, for sufficiently small t_{ter}, $p_{\mathrm{NB}}(t_{\mathrm{ter}}, \boldsymbol{S})$ is nearly equal to 1, since the payoff is paid at least if $\boldsymbol{S}(t)$ does not reach any boundaries and the probability that $\boldsymbol{S}(t)$ reaches any boundaries can be neglected for time close to 0. Besides, note that we obtain the derivative prices only on the points in boundaries by solving PDE. For small t_{ter}, we can approximately calculate V_0 using only the information in boundaries, since the probability distribution of $\boldsymbol{S}(t_{\mathrm{ter}})$ over the boundaries is negligible. In summary, we should set t_{ter} as large as possible in the range of the value for which the probability distribution of $\boldsymbol{S}(t_{\mathrm{ter}})$ is almost confined within the boundaries. For such t_{ter}, we can approximate

$$V_0 \approx e^{-rt_{\mathrm{ter}}} \int_D d\boldsymbol{S}\phi(t_{\mathrm{ter}}, \boldsymbol{S})V(t_{\mathrm{ter}}, \boldsymbol{S}), \tag{4.8}$$

or, equivalently,

$$V_0 \approx e^{-rT} \int_{\tilde{D}} d\boldsymbol{x}\tilde{\phi}(t_{\mathrm{ter}}, \boldsymbol{x})Y(\tau_{\mathrm{ter}}, \boldsymbol{x}), \tag{4.9}$$

where $\tau_{\mathrm{ter}} := T - t_{\mathrm{ter}}$ and $\tilde{\phi}(t, \boldsymbol{x})$ is the probability density of $\boldsymbol{x}(t)$ under the BS model (2.3) and will be explicitly given later.

Considering the above points, we obtain the lemma, which shows a criterion to set t_{ter}. First, we make an assumption, which is necessary to upper bound the contribution from the outside of the boundaries to the integral (4.7).

Assumption 4.3.1. *There exist positive constants $A_0, A_1, ..., A_d$ such that f_{pay} in Problem 1 satisfies*

$$f_{\mathrm{pay}}(\boldsymbol{S}) \leq \sum_{i=1}^{d} A_i S_i + A_0 \tag{4.10}$$

for any $\boldsymbol{S} \in D$.

That is, we assume that the payoff is upper bounded by some linear function, which is the case for many cases such as call/put options on linear combinations of $S_1, ..., S_d$ (i.e. basket options). Then, the following lemma holds.

Lemma 4.3.1. *Consider Problem 1. Under Assumption 4.3.1, for any $\epsilon \in \mathbb{R}_+$ satisfying*

$$\log\left(\frac{\tilde{A}d(d+1)}{\epsilon}\right) > \max\left\{\frac{2}{5}\left(1 - \frac{2r}{\sigma_i^2}\right)\log\left(\frac{U_i}{S_{i,0}}\right), \frac{2}{5}\left(1 - \frac{2r}{\sigma_i^2}\right)\log\left(\frac{S_{i,0}}{L_i}\right)\right\}, i \in [d], \tag{4.11}$$

where $\tilde{A} = \max\{A_1\sqrt{U_1 S_{1,0}}, ..., A_1\sqrt{U_d S_{d,0}}, A_0\}$, *and*

$$\epsilon < 2d(d+1) \times \max\{A_0, A_1 S_{1,0}, ..., A_d S_{d,0}\}, \tag{4.12}$$

the inequality

$$\left| V(0, \boldsymbol{S}_0) - e^{-rT} \int_{\tilde{\tilde{D}}} d\boldsymbol{x} \tilde{\phi}(t_{\text{ter}}, \boldsymbol{x}) Y(t_{\text{ter}}, \boldsymbol{x}) \right| \leq 2\epsilon \tag{4.13}$$

holds, where

$$t_{\text{ter}} := \min \left\{ \frac{2\left(\log\left(\frac{U_1}{S_{1,0}}\right)\right)^2}{25\sigma_1^2 \log\left(\frac{2\tilde{A}d(d+1)}{\epsilon}\right)}, ..., \frac{2\left(\log\left(\frac{U_d}{S_{d,0}}\right)\right)^2}{25\sigma_d^2 \log\left(\frac{2\tilde{A}d(d+1)}{\epsilon}\right)}, \frac{2\left(\log\left(\frac{S_{1,0}}{L_1}\right)\right)^2}{25\sigma_1^2 \log\left(\frac{2\tilde{A}d(d+1)}{\epsilon}\right)}, \right.$$
$$\left. ..., \frac{2\left(\log\left(\frac{S_{d,0}}{L_d}\right)\right)^2}{25\sigma_d^2 \log\left(\frac{2\tilde{A}d(d+1)}{\epsilon}\right)} \right\}. \tag{4.14}$$

and

$$\tilde{\tilde{D}} := \left[\frac{1}{2}\left(l_1 + x_1^{(0)}\right), \frac{1}{2}\left(x_1^{(n_{\text{gr}}-1)} + u_1\right)\right] \times \cdots \times \left[\frac{1}{2}\left(l_d + x_d^{(0)}\right), \frac{1}{2}\left(x_d^{(n_{\text{gr}}-1)} + u_d\right)\right]. \tag{4.15}$$

The proof is given in Appendix A2.2. Note that, in (4.13), the region of the integral is slightly different from $\tilde{D}$, the interior of the boundary in the $\boldsymbol{x}$ domain. This is just for interpreting the finite-sum approximation of the integral as the midpoint rule and explained in the proof of Lemma 4.4.1.

4.4 Quantum method for derivative pricing by FDM

In this section, we finally present the quantum method for derivative pricing by FDM. Our idea is calculating the present derivative price V_0 as (4.7), the expected value of the price at the future time t_{ter}. As explained in Section 4.3, we approximate (4.7) as (4.9). In fact, we have to approximate (4.9) further, since we obtain the derivative prices only on the grid points by solving PDE using FDM. Therefore, we approximate (4.9) as

$$V_0 \approx e^{-rT} \sum_{k=1}^{N_{\text{gr}}} p_k \tilde{Y}_k(\tau_{\text{ter}}), \tag{4.16}$$

where p_k is the existence probability of $\boldsymbol{x}(t_{\text{ter}})$, the log prices of underlying assets at t_{ter}, on the k-th grid point and explicitly defined soon. In other words, we calculate

$$V_0 \approx e^{-rT} \boldsymbol{p} \cdot \tilde{\boldsymbol{Y}}(\tau_{\text{ter}}), \tag{4.17}$$

where $\boldsymbol{p} := (p_1, ..., p_{N_{\text{gr}}})^\top$. Hereafter, we discuss how to estimate this inner product.

4.4.1 Generating the probability vector

Firstly, let us discuss how to generate $\boldsymbol{p}$, a vector which represents $\tilde{\phi}(t_{\rm ter}, \boldsymbol{x})$, the probability distribution of $\boldsymbol{x}(t_{\rm ter})$, as a quantum state. As we will see below, although we aim to generate a quantum state in which the amplitudes of basis states are proportional to $\tilde{\phi}(t_{\rm ter}, \boldsymbol{x})$, we can apply the method to generate a state in which amplitudes are square roots of probabilities[11, 31], since $\tilde{\phi}(t_{\rm ter}, \boldsymbol{x})$ can be regarded as the square roots of the probability densities under another distribution.

Concretely speaking, we aim to generate the vector

$$\begin{aligned} \boldsymbol{p} &:= (p_1, ..., p_{N_{\rm gr}})^\top, \\ p_k &:= \tilde{\phi}(t_{\rm ter}, \boldsymbol{x}) \prod_{i=1}^{d} h_i \end{aligned} \tag{4.18}$$

where $\tilde{\phi}(t, \boldsymbol{x})$, the probability density of $\boldsymbol{x}(t)$, is explicitly given as

$$\begin{aligned} \tilde{\phi}(t, \boldsymbol{x}) &:= \frac{1}{(2\pi t)^{d/2} \left(\prod_{i=1}^{d} \sigma_i \right) \sqrt{\det \rho}} \exp\left(-\frac{1}{2} (\boldsymbol{x} - \boldsymbol{\mu})^\top \Sigma^{-1} (\boldsymbol{x} - \boldsymbol{\mu}) \right), \\ \boldsymbol{\mu} &:= \left(\left(r - \frac{1}{2}\sigma_1^2 \right) t, ..., \left(r - \frac{1}{2}\sigma_d^2 \right) t \right)^\top \\ \Sigma &:= (\sigma_i \sigma_j \rho_{ij})_{\substack{1 \le i \le d \\ 1 \le j \le d}}, \end{aligned} \tag{4.19}$$

that is, the density of the d-dimensional normal distribution with the mean $\boldsymbol{\mu}$ and the covariance matrix Σ. Actually, we generate this vector as a normalized quantum state, that is,

$$\begin{aligned} |\bar{p}\rangle &:= \sum_{k=1}^{N_{\rm gr}} \frac{p_k}{P} |k\rangle, \\ P &:= \|\boldsymbol{p}\| = \sqrt{\sum_{k=1}^{N_{\rm gr}} p_k^2}. \end{aligned} \tag{4.20}$$

Here, note that $(\tilde{\phi}(t, \boldsymbol{x}))^2$ is $\varphi(\boldsymbol{x})$ times a constant independent of $\boldsymbol{x}$, where

$$\varphi(\boldsymbol{x}) := \frac{1}{(\pi t)^{d/2} \left(\prod_{i=1}^{d} \sigma_i \right) \sqrt{\det \rho}} \exp\left(-\frac{1}{2} (\boldsymbol{x} - \boldsymbol{\mu})^\top \left(\frac{1}{2}\Sigma \right)^{-1} (\boldsymbol{x} - \boldsymbol{\mu}) \right), \tag{4.21}$$

is the probability density function for another d-dimensional normal distribution. Therefore, $|\bar{p}\rangle$ is approximately the state $|\varphi\rangle$, where $\varphi(\boldsymbol{x})$ is encoded into the square roots of the amplitudes, that is,

$$|\varphi\rangle := \frac{1}{\sqrt{Q}} \sum_{k=1}^{N_{\rm gr}} \sqrt{q_k} |k\rangle. \tag{4.22}$$

Algorithm 1 Generate $|\bar{p}\rangle$

1: Prepare d $m_{\rm gr}$-qubit registers and initialize all qubits to $|\bar{0}\rangle$, which means the initial state is $\underbrace{|0\rangle \cdots |0\rangle}_{d}$.
2: **for** $i = 1$ to d **do**
3: **for** $j = 1$ to $m_{\rm gr}$ **do**
4: Using $k_1, ..., k_{i-1}$ indicated by the first, ..., $(i-1)$-th registers, respectively, and $k_i^{[1]}, ..., k_i^{[j-1]}$, the bits on the first, ..., $(j-1)$-th qubits of the i-th register, respectively, rotate the j-th qubit in the i-th register as
5:

$$|\bar{0}\rangle \to \sqrt{f_{i,j}(k_1, ..., k_{i-1}; k_i^{[1]}, ..., k_i^{[j-1]})}\,|\bar{0}\rangle + \sqrt{1 - f_{i,j}(k_1, ..., k_{i-1}; k_i^{[1]}, ..., k_i^{[j-1]})}\,|\bar{1}\rangle\,. \tag{4.25}$$

This transforms the entire state into

$$\frac{1}{\sqrt{Q}} \sum_{k_1=0}^{n_{\rm gr}-1} \cdots \sum_{k_{i-1}=0}^{n_{\rm gr}-1} \sum_{k_i^{[1]}=0}^{1} \cdots \sum_{k_i^{[j]}=0}^{1} \sqrt{q_{i,j}(k_1, ..., k_{i-1}; k_i^{[1]}, ..., k_i^{[j]})}\,|k_1\rangle \cdots |k_{i-1}\rangle \left|\tilde{k}\right\rangle \underbrace{|0\rangle \cdots |0\rangle}_{d-i}, \tag{4.26}$$

where $\tilde{k}$ is an integer whose $m_{\rm gr}$-bit representation is $k_i^{[1]} \cdots k_i^{[j]} \underbrace{0 \cdots 0}_{m_{\rm gr}-j}$.

6: **end for**
7: **end for**

Here,

$$q_k := \int_{x_1^{(k_1)}}^{x_1^{(k_1+1)}} dx_1 \cdots \int_{x_d^{(k_d)}}^{x_d^{(k_d+1)}} dx_d \varphi(t_{\rm ter}, \boldsymbol{x}), \text{ for } k = \sum_{i=1}^{d} n_{\rm gr}^{d-i} k_i + 1, k_i = 0, 1, ..., n_{\rm gr} - 1, \tag{4.23}$$

which is close to $\varphi(t_{\rm ter}, \boldsymbol{x}^{(k)}) \prod_{i=1}^{d} h_i$, and

$$Q := \int_{x_1^{(0)}}^{x_1^{(n_{\rm gr}+1)}} dx_1 \cdots \int_{x_d^{(0)}}^{x_d^{(n_{\rm gr}+1)}} dx_d \varphi(t_{\rm ter}, \boldsymbol{x}), \tag{4.24}$$

which is close to 1.

Then, the task is boiled down to generating $|\varphi\rangle$. This can be done by the multivariate extension of the method of [11] for univariate distributions. The concrete procedure is Algorithm 1. Here, note that $|k\rangle$ can be decomposed as

$$|k\rangle = |k_1\rangle \cdots |k_d\rangle\,, \tag{4.27}$$

where each $|k_i\rangle$ is a state on a $m_{\rm gr}$-qubit register (recall that $n_{\rm gr} = 2^{m_{\rm gr}}$), and $|k_i\rangle$ can be further decomposed as

$$|k_i\rangle = \left|\overline{k_i^{[i]}}\right\rangle \cdots \left|\overline{k_i^{[m_{\rm gr}]}}\right\rangle, \tag{4.28}$$

where we write the n-bit representation of $i \in \{0, 1, ..., 2^n - 1\}$ as $i^{[1]} \cdots i^{[n]}$ with $i^{[1]}, ..., i^{[n]} \in \{0, 1\}$. Besides, note that Algorithm 1 requires us to compute

$$f_{i,j}(k_1, ..., k_{i-1}; k_i^{[1]}, ..., k_i^{[j-1]}) := \frac{q_{i,j}(k_1, ..., k_{i-1}; k_i^{[1]}, ..., k_i^{[j-1]}, 0)}{q_{i,j-1}(k_1, ..., k_{i-1}; k_i^{[1]}, ..., k_i^{[j-1]})} \tag{4.29}$$

for $i \in [d]$ and $j \in [m_{\rm gr}]$, where

$$q_{i,j}(k_1, ..., k_{i-1}; b_1, ..., b_j) := \\ \begin{cases} \int_{x_{1,j}^L(b_1,...,b_j)}^{x_{1,j}^R(b_1,...,b_j)} dx_1 \int_{x_2^{(0)}}^{x_2^{(n_{\rm gr}+1)}} dx_2 \cdots \int_{x_d^{(0)}}^{x_d^{(n_{\rm gr}+1)}} dx_d \varphi(t, \boldsymbol{x}) & ;\ i = 1 \\ \int_{x_1^{(k_1)}}^{x_1^{(k_1+1)}} dx_1 \cdots \int_{x_{i-1}^{(k_{i-1})}}^{x_{i-1}^{(k_{i-1}+1)}} dx_{i-1} \int_{x_{i,j}^L(b_1,...,b_j)}^{x_{i,j}^R(b_1,...,b_j)} dx_i & \\ \times \int_{x_{i+1}^{(0)}}^{x_{i+1}^{(n_{\rm gr}+1)}} dx_{i+1} \cdots \int_{x_d^{(0)}}^{x_d^{(n_{\rm gr}+1)}} dx_d \varphi(t, \boldsymbol{x}) & ;\ 2 \le i \le d-1 \\ \int_{x_1^{(k_1)}}^{x_1^{(k_1+1)}} dx_1 \cdots \int_{x_{d-1}^{(k_{d-1})}}^{x_{d-1}^{(k_{d-1}+1)}} dx_{d-1} \int_{x_{d,j}^L(b_1,...,b_j)}^{x_{d,j}^R(b_1,...,b_j)} dx_d \varphi(t, \boldsymbol{x}) & ;\ i = d \end{cases}, \tag{4.30}$$

and

$$x_{i,j}^L(b_1, ..., b_j) := x_i^{(k_L)}, k_L := \begin{cases} 0 & ;\ j = 0 \\ b_1 \cdots b_j \underbrace{0 \cdots 0}_{m_{\rm gr}-j} & ;\ j = 1, ..., m_{\rm gr} \end{cases}$$

$$x_{i,j}^R(b_1, ..., b_j) := x_i^{(k_R)}, k_R := \begin{cases} n_{\rm gr} & ;\ j = 0 \\ b_1 \cdots b_j \underbrace{1 \cdots 1}_{m_{\rm gr}-j} + 1 & ;\ j = 1, ..., m_{\rm gr} \end{cases}, \tag{4.31}$$

for $i \in [d]$, $j = 0, 1, ..., m_{\rm gr}$ and $b_1, ..., b_j \in \{0, 1\}$ (note that $q_{1,0} = Q$). Such a $f_{i,j}$ can be actually computed as follows. Neglecting the contribution from the outside of the boundary, we see that

$$f_{i,j}(k_1, ..., k_{i-1}; k_i^{[1]}, ..., k_i^{[j-1]}) \approx \frac{\int_{x_{i,j}^L(b_1,...,b_j)}^{\frac{1}{2}(x_{i,j}^L(b_1,...,b_j) + x_{1,j}^R(b_1,...,b_j))} dx_i \varphi_i^{\rm mar}(x_i; k_1, ..., k_{i-1})}{\int_{x_{i,j}^L(b_1,...,b_j)}^{x_{i,j}^R(b_1,...,b_j)} dx_i \varphi_i^{\rm mar}(x_i; k_1, ..., k_{i-1})}, \tag{4.32}$$

where

$$\varphi_i^{\mathrm{mar}}(x_i; k_1, ..., k_{i-1}) := \int_{-\infty}^{+\infty} dx_{i+1} \cdots \int_{-\infty}^{+\infty} dx_d \varphi((x_1^{(k_1)}, ..., x_{i-1}^{(k_{i-1})}, x_i, x_{i+1}, ..., x_d)^\top) \tag{4.33}$$

is the marginal density given by integrating out $x_{i+1}, ..., x_d$ and fixing $x_1, ..., x_{i-1}$. We can regard this as an univariate normal distribution density function of x_i (times a constant independent of x_i), and therefore compute (4.32) by the method presented in [31].

At the end of this subsection, let us evaluate the error of (4.17) as an approximation for (4.9). As preparation, we evaluate the normalization factor P as follows:

$$\begin{aligned} P^2 &= \sum_{k=1}^{N_{\mathrm{gr}}} \left(\phi_{\boldsymbol{x}}(t, \boldsymbol{x}_{\mathrm{gr}}^{(k)})\right)^2 \left(\prod_{i=1}^d h_i\right)^2 \\ &\approx \frac{\prod_{i=1}^d h_i}{(4\pi t)^{d/2} \left(\prod_{i=1}^d \sigma_i\right) \sqrt{\det \rho}} \int_{\mathbb{R}^d} d\boldsymbol{x} \varphi(\boldsymbol{x}) \\ &= \frac{\prod_{i=1}^d h_i}{(4\pi t)^{d/2} \left(\prod_{i=1}^d \sigma_i\right) \sqrt{\det \rho}} \\ &\approx \frac{\prod_{i=1}^d \Delta_i}{(4\pi)^{d/2} N_{\mathrm{gr}} \sqrt{\det \rho}}, \end{aligned} \tag{4.34}$$

where

$$\Delta_i := \frac{u_i - l_i}{\sigma_i \sqrt{t_{\mathrm{ter}}}}, i \in [d]. \tag{4.35}$$

Besides, we make an additional assumption.

Assumption 4.4.1. *For $Y(\tau, \boldsymbol{x})$, the solution of (2.20) and (2.21), and $\tilde{\phi}(t, \boldsymbol{x})$, the probability density function of $\boldsymbol{x}(t)$ under the BS model (2.2), there exists $\eta \in \mathbb{R}$ such that*

$$\forall i, j \in [d], \tau \in (0, T), \boldsymbol{x} \in \tilde{D}, \left| \frac{\partial^2}{\partial x_i \partial x_j} (\tilde{\phi}(T - \tau, \boldsymbol{x}) Y(\tau, \boldsymbol{x})) \right| < \eta. \tag{4.36}$$

Then, we obtain the following lemma, which guarantees us that we can approximate the integral by the finite sum over the grid points.

Lemma 4.4.1. *Consider Problem 1. Under Assumptions 2.2.1, 4.3.1 and*

4.4.1, for a given $\epsilon \in \mathbb{R}_+$ satisfying (4.11) and (4.12), if we set

$$h_i < \tilde{h}_i := \min \left\{ \frac{(4\pi)^{d/8}(\det \rho)^{1/8}}{d\sigma_i(\prod_{i=1}^d \Delta_i)^{1/4}} \sqrt{\frac{\epsilon}{2\xi T}}, \right. \\ \left. \frac{(4\pi)^{d/8}(\det \rho)^{1/8}}{\sigma_i(\prod_{i=1}^d \Delta_i)^{1/4}} \sqrt{\frac{\epsilon}{\zeta d T}}, \frac{1}{\left(\prod_{i=1}^d (u_i - l_i)\right)^{1/2}} \sqrt{\frac{24\epsilon}{d\eta}} \right\}, i \in [d] \tag{4.37}$$

the following holds

$$\left| e^{-rT} \boldsymbol{p} \cdot \tilde{\boldsymbol{Y}}(\tau_{\mathrm{ter}}) - V_0 \right| < 4\epsilon, \tag{4.38}$$

where $\boldsymbol{p}$ is defined as (4.18), $\tilde{\boldsymbol{Y}}$ is the solution of (2.23).

The proof is given in Appendix A2.3.

4.4.2 Generating the derivative price vector

Next, let us consider how to generate $\boldsymbol{V}$, the vector which encodes the grid derivative prices at t_{ter}. Precisely speaking, since we solve (2.23), we actually obtain the vector $\tilde{\boldsymbol{Y}}$, which encodes the approximations of $Y(\tau_{\mathrm{ter}}, \boldsymbol{x})$ on the grid points. Furthermore, by the algorithm presented in Section 2.3.2, we obtain not $\tilde{\boldsymbol{Y}}$ itself but some quantum state like (4.5), which contains a state corresponding to $\tilde{\boldsymbol{Y}}$ along with a garbage state.

For the precise discussion, let us firstly make some assumptions in order to satisfy preconditions to use the quantum algorithm. The first one is as follows:

Assumption 4.4.2. *$\boldsymbol{C}(\tau)$ in (2.23) is independent of τ.*

Then, hereafter, we simply write $\boldsymbol{C}(\tau)$ as $\boldsymbol{C}$. We make this assumption in order to fit the current setting to [4], which considered solving (2.43) for constant A and $\boldsymbol{b}$ (note that F in (2.23) is constant). Although $\boldsymbol{C}(\tau)$ is not generally time-independent, the assumption is satisfied in some cases:

- In some cases, a derivative is far in-the-money for a party at some points on the boundary, and this means that the party would receive a constant payoff K at the maturity with high probability. For example,
  - The payoff is the cash-or-nothing type.
  - The payoff is capped, that is, the payoff function takes the form of $f_{\mathrm{pay}}(\boldsymbol{S}) = \min\{f(\boldsymbol{S}), K\}$ with some function $f(\boldsymbol{S})$.

  In these cases, we can approximate that $V(t, \boldsymbol{S}) \approx e^{-r(T-t)}K$, which means that $Y(\tau, \boldsymbol{x}) \approx K$, on the points.

- If a boundary corresponds to a knock-out barrier, $V(t, \boldsymbol{S}) = 0$ on it.

Of course, there are many cases where $\boldsymbol{C}(\tau)$ is time-dependent, and it is desirable to expend our method to such cases. We leave this as a future work.

The second assumption is as follows:

Assumption 4.4.3. *For F in (2.25), the following oracles $O_{F,1}$ and $O_{F,2}$ are available:*

$$O_{F,1} : |j\rangle |l\rangle \mapsto |j\rangle |\nu_F(j,l)\rangle , \tag{4.39}$$

where $j \in [N_{\mathrm{gr}}]$, $l \in [s_F]$, s_F is the sparsity of F, and $\nu(j,l)$ is the column index of the l-th nonzero entry in the j-th row,

$$O_{F,2} : |j\rangle |k\rangle |z\rangle \mapsto |j\rangle |k\rangle |z \oplus F_{jk}\rangle , \tag{4.40}$$

where $j, k \in [N_{\mathrm{gr}}]$ and $z \in \mathbb{R}$. Besides, for $\tilde{\boldsymbol{f}}_{\mathrm{pay}}$ in (2.24) and $\boldsymbol{C}$ in (2.27), we know their norms and the following oracles $O_{\tilde{\boldsymbol{f}}_{\mathrm{pay}}}$ and $O_{\boldsymbol{C}}$ are available:

$$O_{\tilde{\boldsymbol{f}}_{\mathrm{pay}}} : \begin{cases} |\bar{0}\rangle |0\rangle \mapsto \frac{1}{\|\tilde{\boldsymbol{f}}_{\mathrm{pay}}\|} |\bar{0}\rangle \left|\tilde{\boldsymbol{f}}_{\mathrm{pay}}\right\rangle \\ |\bar{1}\rangle |\psi\rangle \mapsto |\bar{1}\rangle |\psi\rangle \quad \text{for any} \quad |\psi\rangle \end{cases} , \tag{4.41}$$

$$O_{\boldsymbol{C}} : \begin{cases} |\bar{0}\rangle |\psi\rangle \mapsto |\bar{0}\rangle |\psi\rangle \quad \text{for any} \quad |\psi\rangle \\ |\bar{1}\rangle |0\rangle \mapsto \frac{1}{\|\boldsymbol{C}\|} |\bar{1}\rangle |\boldsymbol{C}\rangle \end{cases} , \tag{4.42}$$

for $\boldsymbol{C} \neq 0$ and $O_{\boldsymbol{C}}$ is an identity operator for $\boldsymbol{C} = 0$.

Since F is explicitly given as (2.25), the sum of the Kronecker products of tridiagonal matrices, construction of $O_{F,1}$ and $O_{F,2}$ is straightforward. On the other hand, $\tilde{\boldsymbol{f}}_{\mathrm{pay}}$ and $\boldsymbol{C}$ are highly problem-dependent, and so are $O_{\tilde{\boldsymbol{f}}_{\mathrm{pay}}}$ and $O_{\boldsymbol{C}}$. Therefore, we just assume their availability in this chapter, referring to some specific cases. Although the gate complexity preparing a state in which a general vector is amplitude-encoded is exponential in the qubit number [**?**, 94, 95, 96, 97, 98, 99], it can be efficiently performed in the following cases.

- By the analogy with preparation of $|\bar{p}\rangle$, we see that we can prepare $\frac{|\tilde{\boldsymbol{f}}_{\mathrm{pay}}\rangle}{\|\tilde{\boldsymbol{f}}_{\mathrm{pay}}\|}$ if we can efficiently calculate

$$\begin{aligned} &\int_{x_{i,j}^L(b_1,\cdots,b_j)}^{\frac{1}{2}(x_{i,j}^L(b_1,\cdots,b_j)+x_{i,j}^R(b_1,\cdots,b_j))} dx_i \int_{l_i}^{u_i} dx_{i+1} \\ &\cdots \int_{l_d}^{u_d} dx_d \left(\tilde{f}_{\mathrm{pay}}((x_1^{(k_1)}, \cdots, x_{i-1}^{(k_{i-1})}, x_i, x_{i+1}, \cdots, x_d)^\top)\right)^2 \\ &\Big/ \int_{x_{i,j}^L(b_1,\cdots,b_j)}^{x_{i,j}^R(b_1,\cdots,b_j)} dx_i \int_{l_i}^{u_i} dx_{i+1} \cdots \int_{l_d}^{u_d} dx_d \left(\tilde{f}_{\mathrm{pay}}((x_1^{(k_1)}, \cdots, x_{i-1}^{(k_{i-1})}, x_i, x_{i+1}, \cdots, x_d)^\top)\right)^2 , \end{aligned} \tag{4.43}$$

where $x^L_{i,j}$ and $x^R_{i,j}$ are defined as (4.31), for $i \in [d]$, $j \in \{0, 1, \cdots, m_{\mathrm{gr}} - 1\}$, $k_1, \cdots, k_d \in \{0, 1, \cdots, n_{\mathrm{gr}} - 1\}$ and $b_1, \cdots, b_j \in \{0, 1\}$. Although it is difficult to analytically calculate this in general, there are some cases where it is possible. An example is the case where f_{pay} depends on only one underlying asset price, say S_1 (and other assets are relevant to the barrier), and has a simple function form, e.g. the call-option-like $f_{\mathrm{pay}}(\boldsymbol{S}) = \max\{S_1 - K, 0\}$ and the put-option-like $f_{\mathrm{pay}} = \max\{K - S_1, 0\}$. In Appendix A2.6, we consider this case in more detail. We here only note that, in this case, (4.43) can be expressed with elementary functions and approximately calculated by a series of arithmetic operations, which requires only logarithmically many gates with respect to calculation accuracy.

- If all boundaries correspond to knock-out barriers, $\boldsymbol{C} = \boldsymbol{0}$, and therefore $O_{\boldsymbol{C}}$ is just an identity operator.

Then, we obtain the following lemma, whose proof is presented in Appendix A2.4.

Lemma 4.4.2. *Consider the ODE system (2.23). Assume that Assumptions 2.2.1, 4.3.1, 4.4.1, 4.4.2 and 4.4.3 are satisfied. Let ϵ be any positive real number satisfying (4.11) and (4.12), and ϵ' be any positive real number. Then, there exists a quantum algorithm that produces a state $\left|\tilde{\Psi}\right\rangle$ ϵ'-close to*

$$|\Psi\rangle := \frac{1}{\sqrt{\langle\Psi_{\mathrm{gar}}|\Psi_{\mathrm{gar}}\rangle + (p+1)\|\tilde{\boldsymbol{Y}}(\tau_{\mathrm{ter}})\|^2}} \left(|\Psi_{\mathrm{gar}}\rangle + \sum_{j=p(k+1)}^{p(k+2)} |j\rangle \left|\tilde{\boldsymbol{Y}}(\tau_{\mathrm{ter}})\right\rangle \right), \tag{4.44}$$

where $\tilde{\boldsymbol{Y}}(\tau_{\mathrm{ter}})$ is a vector satisfying (4.38), using

$$O\left(\mathcal{C} \times \mathrm{poly}\left(\log\left(\frac{\mathcal{C}}{\epsilon'}\right)\right)\right) \tag{4.45}$$

queries to $O_{F,1}$, $O_{F,2}$, $O_{\tilde{\boldsymbol{f}}_{\mathrm{pay}}}$, and $O_{\boldsymbol{C}}$. Here,

$$\mathcal{C} := \max\left\{ \frac{\sqrt{\prod_{i=1}^d \Delta_i}\, d^2 \Xi \sigma_{\max}^2 \tau_{\mathrm{ter}}}{(4\pi)^{d/4} (\det \rho)^{1/4}}, d\eta \prod_{i=1}^d (u_i - l_i) \right\} \times \frac{\kappa_V d^4 \sigma_{\max}^2 \tau_{\mathrm{ter}}}{\epsilon}, \tag{4.46}$$

$\kappa_V = \|V\| \cdot \|V^{-1}\|$ is the condition number of V which diagonalizes F (i.e. VFV^{-1} is a diagonal matrix), $\sigma_{\max} := \max_{i\in[d]} \sigma_i$, $\Xi := \max\{\xi, \zeta/d\}$, $\tau_{\mathrm{ter}} := T - t_{\mathrm{ter}}$, t_{ter} is defined as (4.14), $p := \lceil \tau_{\mathrm{ter}} \|F\| \rceil$, $k := \lfloor 2 \log \Omega / \log(\log \Omega) \rfloor$, $\Omega = 70 g \kappa_V p^{3/2} (\|\boldsymbol{f}_{\mathrm{pay}}\| + T\|\boldsymbol{C}\|)/\epsilon \|\tilde{\boldsymbol{Y}}(\tau_{\mathrm{ter}})\|$, and $|\Psi_{\mathrm{gar}}\rangle$ is an unnormalized state which takes the form of $|\Psi_{\mathrm{gar}}\rangle = \sum_{j=0}^{p(k+1)-1} |j\rangle |\psi_j\rangle$ with some unnormalized states $|\psi_0\rangle, |\psi_1\rangle, ..., \left|\psi_{p(k+1)-1}\right\rangle$ and satisfies

$$\langle\Psi_{\mathrm{gar}}|\Psi_{\mathrm{gar}}\rangle = O(g^2(p+1)\|\tilde{\boldsymbol{Y}}(\tau_{\mathrm{ter}})\|^2) \tag{4.47}$$

with $g := \max_{\tau\in[0,\tau_{\mathrm{ter}}]} \|\tilde{\boldsymbol{Y}}(\tau)\| / \|\tilde{\boldsymbol{Y}}(\tau_{\mathrm{ter}})\|$.

4.4.3 Proposed algorithm

Finally, based on the above discussions, we present the quantum method to calculate the present derivative price V_0. Our strategy is calculating this as (4.17). More concretely, we aim to subtract the information of $\boldsymbol{p} \cdot \tilde{\boldsymbol{Y}}(\tau_{\mathrm{ter}})$ from $|\Psi\rangle$ in (4.44), the output state of the algorithm of [4].

In order to do this, we first modify the algorithm slightly. That is, we aim to solve not (2.45) but the following one by the QLS algorithm:

$$\tilde{C}_{m,k,p}(Fh_t)\boldsymbol{X} = \boldsymbol{e}_0 \otimes \tilde{\boldsymbol{f}}_{\mathrm{pay}} + h_t \sum_{i=0}^{m-1} \boldsymbol{e}_{i(k+1)+1} \otimes \boldsymbol{C} + \sum_{i=1}^{p+1} \boldsymbol{e}_{m(k+1)+p+i} \otimes \boldsymbol{\gamma}. \tag{4.48}$$

Here, m, p, k are integers defined in the statement of Lemma 4.4.2, $q := m(k+1)+2p+1$, $h_t = \tau_{\mathrm{ter}}/m$, $\boldsymbol{X} \in \mathbb{R}^{N_{\mathrm{gr}}(q+1)}$, $\{\boldsymbol{e}_i\}_{i=0,1,\dots,q}$ is an orthonormal basis of $\mathbb{R}^{q+1}$, and $\boldsymbol{\gamma} := (\gamma, ..., \gamma)^\top \in \mathbb{R}^{N_{\mathrm{gr}}}$ for some $\gamma \in \mathbb{R}_+$. Hereafter, we make the following assumption on γ:

Assumption 4.4.4. *We are given $\gamma \in \mathbb{R}_+$ satisfying*

$$\frac{1}{2}\bar{Y}(\tau_{\mathrm{ter}}) < \gamma < 2\bar{Y}(\tau_{\mathrm{ter}}), \tag{4.49}$$

where

$$\bar{Y}(\tau_{\mathrm{ter}}) := \sqrt{\frac{1}{N_{\mathrm{gr}}} \sum_{k=1}^{N_{\mathrm{gr}}} (Y(\tau_{\mathrm{ter}}, \boldsymbol{x}^{(k)}))^2}. \tag{4.50}$$

This means that γ is comparable with the root mean square of $Y(\tau_{\mathrm{ter}}, \boldsymbol{x})$ on the grid points. Besides, the $N_{\mathrm{gr}}(q+1) \times N_{\mathrm{gr}}(q+1)$ matrix $\tilde{C}_{m,k,p}(Fh_t)$ is now defined as

$$\begin{aligned} \tilde{C}_{m,k,p}(Fh_t) \quad := \quad & \sum_{j=0}^{q} \boldsymbol{e}_j \boldsymbol{e}_j^\top \otimes I_{N_{\mathrm{gr}}} - \sum_{i=0}^{m-1} \sum_{j=1}^{k} \boldsymbol{e}_{i(k+1)+j} \boldsymbol{e}_{i(k+1)+j-1}^\top \otimes \frac{1}{j} Fh_t \\ & - \sum_{i=0}^{m-1} \sum_{j=0}^{k} \boldsymbol{e}_{(i+1)(k+1)} \boldsymbol{e}_{i(k+1)+j}^\top \otimes I_{N_{\mathrm{gr}}} - \sum_{j=m(k+1)+1}^{m(k+1)+p} \boldsymbol{e}_j \boldsymbol{e}_{j-1}^\top \otimes I_{N_{\mathrm{gr}}} \end{aligned} \tag{4.51}$$

,

or, equivalently,

$$\tilde{C}_{m,k,p}(Fh_t) = \begin{pmatrix} C_{m,k,p}(Fh_t) & 0 \\ 0 & I_{N_{\mathrm{gr}}(p+1)} \end{pmatrix}. \tag{4.52}$$

Visually, (4.48) is displayed as follows

$$\begin{pmatrix}
I_{N_{\rm gr}} & & & & & & & & & & & & & & & & & & & & \\
-Fh_t/1 & I_{N_{\rm gr}} & & & & & & & & & & & & & & & & & & \\
 & \ddots & \ddots & & & & & & & & & & & & & & & & & \\
 & & -Fh_t/k & I_{N_{\rm gr}} & & & & & & & & & & & & & & & & \\
-I_{N_{\rm gr}} & \cdots & -I_{N_{\rm gr}} & -I_{N_{\rm gr}} & I_{N_{\rm gr}} & & & & & & & & & & & & & & & \\
 & & & & \ddots & \ddots & & & & & & & & & & & & & & \\
 & & & & & -Fh_t/1 & I_{N_{\rm gr}} & & & & & & & & & & & & & \\
 & & & & & & \ddots & \ddots & & & & & & & & & & & & \\
 & & & & & & & -Fh_t/k & I_{N_{\rm gr}} & & & & & & & & & & & \\
 & & & & & -I_{N_{\rm gr}} & \cdots & -I_{N_{\rm gr}} & -I_{N_{\rm gr}} & I_{N_{\rm gr}} & & & & & & & & & & \\
 & & & & & & & & & -I_{N_{\rm gr}} & I_{N_{\rm gr}} & & & & & & & & & \\
 & & & & & & & & & & \ddots & \ddots & & & & & & & & \\
 & & & & & & & & & & & -I_{N_{\rm gr}} & I_{N_{\rm gr}} & & & & & & & \\
 & & & & & & & & & & & & I_{N_{\rm gr}} & & & & & & & \\
 & & & & & & & & & & & & & \ddots & & & & & & \\
 & & & & & & & & & & & & & & I_{N_{\rm gr}} & & & & &
\end{pmatrix} \boldsymbol{X} = \begin{pmatrix} \tilde{\boldsymbol{f}}_{\rm pay} \\ h_t \boldsymbol{C} \\ 0 \\ \vdots \\ 0 \\ \vdots \\ h_t \boldsymbol{C} \\ 0 \\ \vdots \\ 0 \\ 0 \\ \vdots \\ 0 \\ \boldsymbol{\gamma} \\ \vdots \\ \boldsymbol{\gamma} \end{pmatrix}. \tag{4.53}$$

The solution of (4.48) is

$$\boldsymbol{X} = \sum_{i=0}^{m-1} \sum_{j=1}^{k} \boldsymbol{e}_{i(k+1)+j} \otimes \tilde{\tilde{\boldsymbol{Y}}}_{i,j} + \sum_{j=0}^{p} \boldsymbol{e}_{m(k+1)+j} \otimes \tilde{\tilde{\boldsymbol{Y}}}(\tau_{\rm ter}) + \sum_{j=1}^{p+1} \boldsymbol{e}_{m(k+1)+p+j} \otimes \boldsymbol{\gamma}, \tag{4.54}$$

for some vectors $\tilde{\tilde{\boldsymbol{Y}}}_{i,j}, \tilde{\tilde{\boldsymbol{Y}}}(\tau_{\rm ter}) \in \mathbb{R}^{N_{\rm gr}}$, and $\tilde{\tilde{\boldsymbol{Y}}}(\tau_{\rm ter})$ becomes close to $\tilde{\boldsymbol{Y}}(\tau_{\rm ter})$. Note that, in $\boldsymbol{X}$, $\tilde{\tilde{\boldsymbol{Y}}}(\tau_{\rm ter})$ and $\boldsymbol{\gamma}$ are repeated $(p+1)$-times. Then, applying the quantum algorithm, we can generate the quantum state $\left|\tilde{\Psi}_{\rm mod}\right\rangle$ ϵ-close to

$$\begin{aligned} |\Psi_{\rm mod}\rangle &:= \frac{1}{Z}\left(|\Psi_{\rm gar}\rangle + \sum_{j=p(k+1)}^{p(k+2)} |j\rangle \left|\tilde{\boldsymbol{Y}}(\tau_{\rm ter})\right\rangle + \sum_{j=p(k+2)+1}^{p(k+3)+1} |j\rangle\,|\boldsymbol{\gamma}\rangle \right), \\ Z &:= \sqrt{\langle\Psi_{\rm gar}|\Psi_{\rm gar}\rangle + (p+1)\|\tilde{\boldsymbol{Y}}(\tau_{\rm ter})\|^2 + (p+1)N_{\rm gr}\gamma^2}. \end{aligned} \tag{4.55}$$

Note that the query complexity for generating $\left|\tilde{\Psi}_{\rm mod}\right\rangle$ is (4.45), similarly to $\left|\tilde{\Psi}\right\rangle$. This is because the complexity of the QLS algorithm depends only on the condition number and sparsity of the matrix and the tolerance[23], and the condition number and sparsity of $\tilde{C}_{m,k,p}(Fh_t)$ is same as $C_{m,k,p}(Fh_t)$.

Using $\left|\tilde{\Psi}_{\rm mod}\right\rangle$, we can estimate $\boldsymbol{p}\cdot\tilde{\boldsymbol{Y}}(\tau_{\rm ter})$. The outline is as follows. First, we estimate the inner product

$$\langle\Psi_{\rm mod}|\Pi|\Psi_{\rm mod}\rangle = \frac{\sqrt{p+1}}{PZ}\boldsymbol{p}\cdot\tilde{\boldsymbol{Y}}(\tau_{\rm ter}), \tag{4.56}$$

where

$$|\Pi\rangle := \frac{1}{\sqrt{p+1}} \sum_{j=p(k+1)}^{p(k+2)} |j\rangle\,|\bar{p}\rangle\,, \tag{4.57}$$

Algorithm 2 Calculate $e^{-r\tau_{\mathrm{ter}}}\boldsymbol{p}\cdot\tilde{\boldsymbol{Y}}(\tau_{\mathrm{ter}})$

Require:

1: $\gamma\in\mathbb{R}_+$ satisfying (4.49).
2: $\epsilon\in\mathbb{R}_+$ satisfying (4.11) and (4.12).
3: $\epsilon_1\in\mathbb{R}_+$ satisfying (4.61).
4: $\epsilon_2\in\mathbb{R}_+$ satisfying (4.62).
5: $\epsilon_{\tilde{\Psi}_{\mathrm{mod}}}\in\mathbb{R}_+$ satisfying (4.65).
6: Accesses to the oracle $U_{\tilde{\Psi}_{\mathrm{mod}}}$ such that (4.58) and (4.64) and its inverse.
7: Accesses to the oracle U_{Π} such that (4.59) and its inverse.
8: Estimate the amplitude of $|0\rangle|0\rangle$ in the state $U_{\Pi}^{\dagger}U_{\tilde{\Psi}_{\mathrm{mod}}}|0\rangle|0\rangle$ by QAE with tolerance ϵ_1. Let the output be E_1.
9: Estimate the square root of the probability that we obtain either of $p(k+2)+1,...,p(k+3)+1$ when we measure the first register of $\left|\tilde{\Psi}_{\mathrm{mod}}\right\rangle$ by QAE with tolerance ϵ_2. Let the output be E_2.
10: Output $e^{-r\tau_{\mathrm{ter}}}\frac{\gamma\sqrt{N_{\mathrm{gr}}}PE_1}{E_2}=:\omega$, where P is given by (4.34).

by estimating the amplitude of $|0\rangle|0\rangle$ in $U_{\Pi}^{\dagger}U_{\tilde{\Psi}_{\mathrm{mod}},\epsilon}|0\rangle|0\rangle$ using QAE. Here, $U_{\tilde{\Psi}_{\mathrm{mod}}}$ and U_{Π} are the unitary operators such that

$$U_{\tilde{\Psi}_{\mathrm{mod}}}|0\rangle|0\rangle=\left|\tilde{\Psi}_{\mathrm{mod}}\right\rangle, \tag{4.58}$$

and

$$U_{\Pi}|0\rangle|0\rangle=|\Pi\rangle, \tag{4.59}$$

respectively. Note that, if we can generate $|\bar{p}\rangle$, we can also generate $|\Pi\rangle$, since this is just a tensor product of $\frac{1}{\sqrt{p+1}}\sum_{j=p(k+1)}^{p(k+2)}|j\rangle$ and $|\bar{p}\rangle$. Next, by QAE, we estimate the probability that we obtain $j\in\{p(k+2)+1,...,p(k+3)+1\}$ in the first register when we measure $\left|\tilde{\Psi}_{\mathrm{mod}}\right\rangle$, and then obtain an estimation of $\gamma\sqrt{(p+1)N_{\mathrm{gr}}}/Z$. Finally, using E_1 and E_2, the outputs of the first and second estimations, respectively, we calculate

$$\frac{e^{-rT}\gamma\sqrt{N_{\mathrm{gr}}}PE_1}{E_2} \tag{4.60}$$

as an estimation of $e^{-rT}\boldsymbol{p}\cdot\tilde{\boldsymbol{Y}}(\tau_{\mathrm{ter}})$. We present the detailed procedure is described as Algorithm 2. Here, taking some $\epsilon\in\mathbb{R}_+$, we require the tolerances ϵ_1 and ϵ_2 in calculating E_1 and E_2 be

$$\epsilon_1 = O\left(\frac{(2\pi)^{d/2}\sqrt{\det\rho}\epsilon}{g\left(\prod_{i=1}^{d}\Delta_i\right)\bar{V}}\right), \tag{4.61}$$

$$\epsilon_2 = O\left(\frac{\epsilon}{gV_0}\right) \tag{4.62}$$

respectively, where

$$\begin{aligned} \bar{V}(t_{\mathrm{ter}}) &:= \sqrt{\frac{1}{N_{\mathrm{gr}}}\sum_{k=1}^{N_{\mathrm{gr}}}(V(t_{\mathrm{ter}},\boldsymbol{S}^{(k)}))^2}, \\ \boldsymbol{S}^{(k)} &:= (S_1^{(k)},...,S_d^{(k)})^\top := (\exp\left(x_1^{(k_1)}\right),...,\exp\left(x_d^{(k_d)}\right))^\top \\ &\quad \text{for } k=\sum_{i=1}^{d} n_{\mathrm{gr}}^{d-i}k_i+1, k_i=0,1,...,n_{\mathrm{gr}}-1 \end{aligned} \tag{4.63}$$

is the root mean square of the derivative prices on the grid points at time t_{ter}. Besides, we require that

$$\| \left|\tilde{\Psi}_{\mathrm{mod}}\right\rangle - |\Psi_{\mathrm{mod}}\rangle \| < \epsilon_\Psi, \tag{4.64}$$

where

$$\epsilon_\Psi = O\left(\max\{\epsilon_1,\epsilon_2\}\right). \tag{4.65}$$

These requirements guarantee the overall error to be smaller than ϵ. We formally state these points along with the complexity of the procedure in Theorem 4.4.1, whose proof is presented in Appendix A2.5.

Theorem 4.4.1. *Consider Problem 1. Assume that Assumptions 2.2.1, 4.3.1, 4.4.1, 4.4.2, 4.4.3 and 4.4.4 are satisfied. Then, for any $\epsilon \in \mathbb{R}_+$ satisfying (4.11) and (4.12), Algorithm 2 outputs the real number ω such that*

$$|\omega - V_0| = O(\epsilon) \tag{4.66}$$

with a probability higher than a specified value (say, 0.99). In this procedure,

$$O\left(\mathcal{D}\times \mathrm{poly}\left(\log \mathcal{D}\right)\right) \tag{4.67}$$

queries to $O_{F,1}$, $O_{F,2}$, $O_{\tilde{\boldsymbol{f}}_{\mathrm{pay}}}$, and $O_{\boldsymbol{C}}$, where

$$\begin{aligned} \mathcal{D} := \max\left\{\frac{\sqrt{\prod_{i=1}^d \Delta_i}d^2\Xi\sigma_{\max}^2\tau_{\mathrm{ter}}}{(4\pi)^{d/4}(\det\rho)^{1/4}}, d\eta\prod_{i=1}^d(u_i-l_i)\right\} \\ \times \max\left\{\frac{\left(\prod_{i=1}^d\Delta_i\right)\bar{V}}{(2\pi)^{d/2}\sqrt{\det\rho}}, V_0\right\} \times \frac{g\kappa_V d^4\sigma_{\max}^2\tau_{\mathrm{ter}}}{\epsilon^2}, \end{aligned} \tag{4.68}$$

$\sigma_{\max} := \max_{i\in[d]}\sigma_i$, $\Xi := \max\{\xi,\zeta/d\}$, $\tau_{\mathrm{ter}} := T - t_{\mathrm{ter}}$, t_{ter} is defined as (4.14), Δ_i is defined as (4.35), $g := \max_{\tau\in[0,\tau_{\mathrm{ter}}]}\|\tilde{\boldsymbol{Y}}(\tau)\|/\|\tilde{\boldsymbol{Y}}(\tau_{\mathrm{ter}})\|$, $\bar{V}(t_{\mathrm{ter}}) := \sqrt{\frac{1}{N_{\mathrm{gr}}}\sum_{k=1}^{N_{\mathrm{gr}}}(V(t_{\mathrm{ter}},\boldsymbol{S}^{(k)}))^2}$, and $\kappa_V = \|V\|\cdot\|V^{-1}\|$ is the condition number of V, which diagonalizes F.

Let us make some comments. First, note that the upper bound of the complexity (4.67) does not have any factor like $(1/\epsilon)^{\mathrm{poly}(d)}$, which means the tremendous speedup with respect to ϵ and d compared with the classical FDM. On the other hand, the exponential dependence on d has not completely disappeared. In fact, (4.67) contains some constants to the power of d, and factors such as $\prod_{i=1}^{d}(u_i - l_i)$ and $\prod_{i=1}^{d}\Delta_i$, that is, the d-times product of $u_i - l_i$ or Δ_i. Recall that $u_i - l_i = \log(U_i/L_i)$ is the width between boundaries in the direction of x_i, the logarithm of the i-th underlying asset price, and Δ_i is that divided by $\sigma_i\sqrt{t_{\mathrm{ter}}}$, which roughly measures the extent of the probability distribution of x_i at time t_{ter}. Therefore, these factors are just logarithmic factors to the power of d. Also note that, in (4.67), there is a factor of $O(d^6)$, which is polynomial but rather strongly dependent on d.

Second, we note that some calculation parameters are difficult to be determined in advance of pricing. For example, although we have assumed that we know γ such that (4.49) holds in advance, it is difficult because we do not know $\bar{Y}(\tau_{\mathrm{ter}})$. Besides, although we set $p = \lceil \|F\|\tau_{\mathrm{ter}} \rceil$ in using the algorithm of [4], it is difficult to set p to this specific value since we can upper bound $\|F\|$ but cannot calculate it precisely. Even in [4], the way to set $p = \lceil \|F\|\tau_{\mathrm{ter}} \rceil$ is not presented. Similar discussion can be applied to other parameters: k, h_i, and so on. Fortunately, the algorithm works not only for such specific values of the parameters but also for comparable values. The factor 1/2 and 2 in (4.49) can be replaced with comparable values (say, 1/3 and 3), which results in change of the complexity only by some $O(1)$ factor. p larger than but comparable with $\lceil \|F\|\tau_{\mathrm{ter}} \rceil$ (say, $2\lceil \|F\|\tau_{\mathrm{ter}} \rceil$) results in comparable computational accuracy and complexity with those for $p = \lceil \|F\|\tau_{\mathrm{ter}} \rceil$. In reality, we may perform computation for various parameter values and search the appropriate ranges of the parameters, for which the calculated derivative price seems to converge. In the practical business, once we find a set of appropriate calculation parameters, we can continue to use it with periodic check of convergence, since we typically perform pricing many times in different but similar settings on model parameters (e.g. σ_i) and contract terms (e.g. barrier level).

4.4.4 Toffoli count estimation in a concrete example

By now, we have evaluated the oracle call number in the proposed algorithm using big-O notation. Now, taking a simple concrete problem setting and assuming some typical values of parameters, we estimate the time complexity of the proposed algorithm, based on the resource estimation for the oracle implementation. This will help to assess the feasibility of actual applications in the future. We measure time complexity by counting the number of costly gates in the implementation. Concretely, we take the number of Toffoli gates (the Toffoli count) as a metric, as [59]. The concrete setting we consider is as follows. As a pricing target, we consider the following derivative contract.

It refers to d underlying asset prices $\boldsymbol{S}(t) = (S_1(t), \ldots, S_d(t))^\top$. The payoff function depends on only S_1 like a put option: $f_{\text{pay}}(\boldsymbol{S}) = \max\{K - S_1, 0\}$ with a strike $K \in \mathbb{R}_+$. All assets including S_1 are involved in the barrier condition. For every $i \in [d]$, two barriers L_i and U_i such that $L_i < S_{i,0} < U_i$ are set for the ith underlying asset, and, for $i = 1, L_1 < K < U_1$ is also required. Then, the option expires with no payoff if either of asset prices reaches either of its barriers by the maturity T, and, otherwise, the above payoff occurs. That is, for every underlying asset, *double knock-out barriers* are set. Such double knock-out options are often traded in the actual financial market for the single asset case, $d = 1$. In this setting, the state $\left|\tilde{\boldsymbol{f}}_{\text{pay}}\right\rangle / \|\tilde{\boldsymbol{f}}_{\text{pay}}\|$ in which the payoff is amplitude encoded can be generated as described in Appendix A2.6. Besides, as mentioned in Section 4.4.1, $O_{\boldsymbol{C}}$ is now an identity operator, and therefore, we will neglect it hereafter. Then, we can estimate the Toffoli count as follows. In the proposed algorithm, $U_{\tilde{\Psi}_{\text{mod}}}$ is iteratively called in QAE, and, also inside $U_{\tilde{\Psi}_{\text{mod}}}$, the oracles $O_{F,1}$, $O_{F,2}$, and $O_{\tilde{\boldsymbol{f}}_{\text{pay}}}$ are repeatedly called. Among them, $O_{\tilde{\boldsymbol{f}}_{\text{pay}}}$, which corresponds to generation of the state $\left|\tilde{\boldsymbol{f}}_{\text{pay}}\right\rangle / \|\tilde{\boldsymbol{f}}_{\text{pay}}\|$, contains many time-consuming computations such as exponential and arcsin, as explained in the Appendix A2.6, and, therefore, makes a dominant contribution to time complexity. Although U_Π is called iteratively in QAE and corresponds to generation of the state $|\bar{p}\rangle$, it is just one state generation and, therefore, makes a smaller contribution than many calls to $O_{\tilde{\boldsymbol{f}}_{\text{pay}}}$ in $U_{\tilde{\Psi}_{\text{mod}}}$. Consequently, using the estimate of Toffoli count of $O_{\tilde{\boldsymbol{f}}_{\text{pay}}}$ pay in the Appendix A2.6 we can estimate the total Toffoli count of the proposed algorithm as $n_{\text{Tof},1} := 1.3 \times 10^4 \times m_{\text{gr}}$ times (4.67). As elaborated in Appendix A2.6, the main contribution to the Toffoli count comes from calculation of arcsin.

Let us get a concrete value for it, assuming reasonable values of parameters. We replace the symbols in (4.67) and (4.68) as follows.

- We set
$$\Delta_i = \frac{u_i - l_i}{\sigma_i \sqrt{t_{\text{ter}}}} = 5\sqrt{2\log\left(\frac{Kd(d+1)}{\epsilon}\right)} \tag{91}$$
for every $i \in [d]$. Note that this is the infimum of Δ_i under (4.14) with $\tilde{A} = K$. Also note that we can set $\tilde{A} = K$ because $f_{\text{pay}}(\boldsymbol{S}) = \max\{K - S_1, 0\} \leq K$.

- We set $u_i - l_i = 1$ for every $i \in [d]$. This means that the ratio of the upper and lower barrier is $U_i/L_i = e \sim 2.7$, which is conceivable in an actual contract.

- We set $\epsilon/K = 0.01$, which corresponds to the 1% price error tolerance relative to the strike.

d	Toffoli-count
1	6.2×10^9
2	9.8×10^{12}
3	2.8×10^{15}
4	3.4×10^{17}
5	3.4×10^{19}

Table 4.1: The estimates of the Toffoli-count of the proposed algorithm for various d in the setting described above.

- We set

$$h_i = \frac{1}{d\sigma_{\max}\sqrt{T}}\sqrt{\frac{3\epsilon}{2\Xi}}, \tag{4.69}$$

  which satisfies (2.29).

- We set $\sigma_{\mathrm{mar}}T = 1$, which is conceivable for years-long derivatives in the actual market [1].

- We replace $\bar{V}$ and V_0 with K. This in fact contributes to the estimation conservatively, since $\bar{V}$ and V_0 are smaller than K, which follows from the fact that a payoff larger than K never arises in the contract.

- We also replace Ξ and η with K, expecting the moderate variation of the derivative price over the space of underlying asset prices[7].

- We also expect the moderate variation of the derivative price over time, and set $g = 1$ for simplicity.

- We expect that underlying asset prices have moderate correlations and that therefore $\det \rho$ is not apart from 1 so much. We now set $\det \rho = 1$ for simplicity.

- Since the leading part with respect to $1/h_i$ in F is symmetric, we expect that κ_V is close to 1. Note that, when F is symmetric, V is unitary and therefore $\kappa_V = 1$. We now set $\kappa_V = 1$ for simplicity.

Then, we obtain the estimate for the total Toffoli-count. In Table 1, we show the value of $n_{\mathrm{Tof},1}\mathcal{D}$ for $d = 1$ to 5, omitting the $O(\mathrm{polylog}(\mathcal{D}))$ factor in (4.66).

[7] In fact, the payoff function, which is the initial condition of the current PDE solving, often has several nonsmooth points, e.g. the strike for a call/out option. Fortunately, there are some techniques to cope with such points and suppress the numerical errors from them, such as tuning placement of grids [100, 101]. In this chapter, we do not go into such technical details but simply assume that this issue is appropriately handled.

As this shows, we need a huge number of Toffoli gates even for the moderate underlying asset number d and the error tolerance ϵ/K. Considering the resent estimation $170\mu sec$ for the runtime of one Toffoli gate in a fault-torelant machine [102], the runtime of the proposed algorithm is formidably long. Although, in theory, the proposed method eventually surpasses the classical method with the time complexity of $O((1/\epsilon)^{\mathrm{poly}(d)})$ for a small ϵ and a large d, it might be difficult to provide some practical quantum advantage. This is similar to the discussion in [102]

4.5 Conclusion

In this chapter, we studied how to apply the quantum algorithm of [4] for solving linear differential equations to pricing multi-asset derivatives by FDM. As we explained, FDM is an appropriate method for pricing some types of derivatives such as barrier options, but suffers from the so-called curse of dimensionality, which makes FDM infeasible for large d, the number of underlying assets, since the dimension of the corresponding ODE system grows as $\epsilon^{\mathrm{poly}(d)}$ for the tolerance ϵ, and so does the complexity. We saw that the quantum algorithm for solving ODE systems, which provides the exponential speedup with respect to the dimensionality compared with classical methods, is beneficial also for derivative pricing. In order to address the specific issue for derivative pricing, that is, extracting the present price from the output state of the quantum algorithm, we adopted the strategy that we calculate the present price as the expected value of the price at some appropriate future time t_{ter}. Then, we constructed the concrete calculation procedure, which is combination of the algorithm of [4] and QAE. We also estimated the query complexity of our method, which does not have any dependence like $(1/\epsilon)^{\mathrm{poly}(d)}$ and shows tremendous speedup with respect to ϵ and d.

We believe that this algorithm is the first step for the research in this direction, but there remains many points to be improved. First, we should consider whether the assumptions we made can be mitigated. For example, although we assume that $\boldsymbol{C}(\tau)$ is time-independent (Assumption 4.4.2), some products do not fit to this condition: e.g., when we consider the upper boundary condition in the case of the European-call-like payoff $f_{\mathrm{pay}}(S) = \max\{S - K, 0\}$ with some constant K, $V(t, S) \approx S - e^{-r(T-t)}K$ and therefore $Y(\tau, \boldsymbol{x}) = e^{r\tau}V(t, \boldsymbol{S})$ cannot be regarded as constant for large S. In order to omit this assumption, we might be able to extend the algorithm of [4] so that it can be applied to time-dependent $\boldsymbol{C}(\tau)$[8]

[8] Actually, the algorithm in [83], which is based on the spectral method, can deal with time-dependent $\boldsymbol{C}(\tau)$. However, in order to apply this algorithm, $V(t, \boldsymbol{S})$ must be smooth enough in the direction of t. On the other hand, in practice, the BS model parameters are often not smooth: for example, piece-wise constant volatilities are often used, which

Another important aspect is pricing early-exercisable derivatives. American-type (resp. Bermudan-type) derivatives, in which either of parties can terminate the contract at any time (resp. at either of some predetermined dates) before the final maturity T, are widely traded and their pricing is important for banks. FDM is suitable and often used for pricing such products, since it determines the derivative price backward from T and can take into account early exercise. However, it is not straightforward to apply the quantum method proposed in this chapter to pricing early-exercisable products. This is because, at exercisable date $t_{\rm exe}$, we need the operation $V(t_{\rm exe}, \boldsymbol{S}) = \max\{V(t_{\rm exe}+0, \boldsymbol{S}), f_{\rm pay}(\boldsymbol{S})\}$, where $V(t_{\rm exe}+0, \boldsymbol{S})$ is the derivative price right after $t_{\rm exe}$, but nonlinear operations on amplitudes such as the max function cannot be implemented on a quantum computer naively.

Including these points, we will investigate the possibility that the quantum FDM speedups pricing for the wider range of derivatives with fewer resources in the future work.

---

deteriorates smoothness of $V(t, \boldsymbol{S})$. In such a case, the algorithm of [4] is expected to be more suitable than that of [83], since the formal solution (2.44) is valid also for piece-wise constant model parameters. That is, if

$$A = \begin{cases} A_1 & ; \ 0 \leq t \leq t_{\rm dis} \\ A_2 & ; \ t_{\rm dis} < t \leq T \end{cases}, \boldsymbol{b} = \begin{cases} \boldsymbol{b}_1 & ; \ 0 \leq t \leq t_{\rm dis} \\ \boldsymbol{b}_2 & ; \ t_{\rm dis} < t \leq T \end{cases},$$

with some $t_{\rm dis} \in (0, T)$, $\boldsymbol{x}(t)$ can be written as

$$\boldsymbol{x}(t) = \begin{cases} e^{A_1 t}\boldsymbol{x}_{\rm ini} + (e^{A_1 t} - I_N)A_1^{-1}\boldsymbol{b}_1 & ; \ 0 \leq t \leq t_{\rm dis} \\ e^{A_2(t-t_{\rm dis})}\boldsymbol{x}(t_{\rm dis}) + (e^{A_2(t-t_{\rm dis})} - I_N)A_2^{-1}\boldsymbol{b}_2 & ; \ t_{\rm dis} < t \leq T \end{cases}.$$

Chapter 5

Pricing of multi-asset derivative with variational quantum simulation

In Sec. 4, we discuss the quantum algorithm for pricing multi-asset derivative by FDM. The algorithm is constructed on the QLSA and thus requires the FTQC. To make the algorithm feasible to run on a small quantum computer, we use variational quantum algorithm. This chapter present the variational quantum algorithm for pricing multi-asset derivatives by FDM based on [K.Kubo, K.Miyamoto, K.Mitarai, K.Fujii, arXiv:2207.01277] and slightly modified to fit the context.

5.1 Introduction

Quantum computers actively utilize quantum phenomena to solve large-scale problems that could not be performed with conventional classical computers. In recent years, applications of quantum computers have been discussed in financial engineering. Specifically, the applications include portfolio optimization [76, 74, 75], risk measurement [103, 71, 72, 104, 73], and derivative pricing [19, 63, 64, 65, 29, 28, 27, 105, 67, 68, 69, 31, 106, 30]. Comprehensive reviews of these topics are presented in Refs. [78, 77, 79, 107].

Among these applications, we consider the pricing of derivatives. Derivatives are the products that refer to the prices of underlying assets such as stocks, bonds, currencies, etc., and their payoff depends on the prices of the assets. For example, a European call option, one of the simplest derivatives, has a predetermined maturity $T > 0$ and strike price K, and its holder gets paid back $\max(S(T) - K, 0)$ for the asset price $S(T)$ at T. For such a simple derivative, the theoretical price can be computed analytically in some models such as the Black-Scholes (BS) model [35]. If one wishes to calculate prices for derivatives with more complex payoffs, numerical calculations are

required [1].

There are many algorithms for numerical calculations. For the pricing of certain types of derivatives, such as barrier options, it is suitable to solve the partial differential equations (PDE) called Black-Scholes PDE (BSPDE) [2] by discretizing them using the finite difference method (FDM). However, in the case of multi-asset derivatives, the number of grid points increases exponentially with respect to the number of referenced assets, making price calculation difficult. When the number of assets is d and the number of grid points is n_{gr} for one asset, the total number of grid points is n_{gr}^d. If we take n_{gr} in proportion to $\epsilon^{-1/2}$ to achieve the error level ϵ (see Lemma II.1 in [108]), classical FDM requires the computational complexity of $O((1/\epsilon)^{O(d)})$.

To overcome this difficulty, several methods [29, 27, 28, 30] have been proposed to efficiently solve the BSPDE using quantum computers. However, when solving the discretized BSPDE with these quantum algorithms, the target derivative price is embedded in the amplitude of one basis of the resulting quantum state, so it requires exponentially large computational complexity to extract it as classical information. Ref. [108] has shown that the complexity can be substantially reduced using the fact that the present derivative price can be calculated as the expected value of the discounted derivative price at a future point in time. They calculate the inner product of the state in which the future derivative prices are embedded and the state in which the probability distribution is embedded using the quantum amplitude estimation (QAE) [6]. Instead of retrieving one of the amplitudes of the output state of the quantum algorithm, the present price of the derivative can be efficiently calculated since all of the amplitudes can be used. In fact, the complexity of the method proposed in Ref. [108] does not have a factor like $(1/\epsilon)^{O(d)}$, but has only $\mathrm{poly}(1/\epsilon, d)$. This means that their method has substantial speedup compared to the classical FDM.

However, it should be noted that their method is constructed on the quantum ordinary differential equation (ODE) solver [4] and the QAE, which requires a large-scale quantum computer with error correction. In addition, it is assumed that we are given the oracle that generates a quantum state in which the boundary conditions of the BSPDE are encoded in amplitudes. As the derivatives are currently dealt with in practice, it is desirable to calculate derivative prices even with a small-scale quantum computer closer to realization.

In this chapter, we propose a variational quantum algorithm for pricing multi-asset derivatives. This is the way to exploit the essential feature proposed in Ref. [108] with variational quantum algorithms and hence thought to work with near-term quantum computers. Our algorithm has the following three parts; embedding the probability distribution of the underlying asset prices into the quantum state, solving the BSPDE with boundary conditions, and calculating the inner product. For the first part, we can use

the quantum generative algorithms [15, 14, 12, 13, 17] or variational quantum simulation (VQS) for the Fokker-Planck equations [5, 24, 9, 70, 30], which describe the time evolution of the probability density functions of the stochastic processes. For the second part, we discretize the BSPDE using the FDM and solve it using VQS. For the third part, we evaluate the square of the inner product of the states, obtained by the first and the second parts of our method, using the SWAP test [109]. Taking the square root of the output of the SWAP test and discounting by the interest rate, we obtain the present price of the derivative. Although there is no guarantee of overall computational complexity due to the heuristic nature of the variational algorithm, we show that the number of measurements of the SWAP test has no factors like $(1/\epsilon)^{O(d)}$, which means that our method can avoid the bottleneck of retrieving derivative prices from the quantum state. Since our algorithm requires quantum circuits with $O(\mathrm{poly}(d\log(1/\epsilon))$ few-qubit gates, even a small-scale quantum computer would be able to perform derivatives pricing with our method. We perform numerical calculations for a single asset double barrier option and confirm that our method is feasible.

This chapter is organized as follows. Sec. 5.2 is the preliminary section. We summarize the related works in Sec. 5.2.1. In Sec. 5.2.2 we introduce derivative pricing using the BSPDE with boundary conditions. We also introduce FDM to discretize the BSPDE and obtain an ODE in Sec. 5.2.3. Sec. 2.3.3 gives an introduction to VQS, which is an algorithm for solving the ODE. In Sec. 5.3, we describe the proposed method. We estimate the number of measurements required by the SWAP test in Sec. 5.3.1 and the whole time complexity of the proposed method in Sec. 5.3.2. We show the feasibility of our method through numerical simulations in Sec. 5.4. Conclusions are given in Sec. 5.5.

5.2 Preliminary

5.2.1 Related work

In this subsection, we explain the existing algorithms for solving the BSPDE with quantum computers. Ref. [27] transforms the BSPDE into a Schrödinger equation, discretize the Hamiltonian by FDM, and solves it by diagonalization of discretized momentum operator with a quantum Fourier transformation. Refs. [28, 29, 30] solve the discretized Schrödinger equation by VQS, which is a variational quantum algorithm for solving ODEs. In the previous studies mentioned above, the time complexity required to solve the BSPDE depends on the grid points only logarithmically. However, there is still a problem that cannot be overlooked; extracting the calculated result from quantum computers may take exponentially long time with respect to d. Solving the BSPDE from the maturity ($t = T$) to the present ($t = 0$) with these quantum algorithms yields unnormalized state $|\boldsymbol{V}(0)\rangle$ whose elements

are the derivative prices on the grid points of underlying asset prices. Note that, typically, we are interested in only one element of $|\boldsymbol{V}(0)\rangle$, the derivative price on the grid point corresponding to the present underlying asset prices. However, since $|\boldsymbol{V}(0)\rangle$ has $O((1/\epsilon)^{O(d)})$ elements, the amplitude corresponding to V_0 in (normalized) $|\boldsymbol{V}(0)\rangle$ is exponentially small. Therefore, the exponential time complexity is required to retrieve V_0 as classical information, and the quantum speedup will be lost.

Ref. [108] shows the algorithm to overcome the problem. They prepare the state $|\boldsymbol{p}(t_{\mathrm{ter}})\rangle$ in which the probability distribution of underlying asset prices on the grid points at a certain time $t_{\mathrm{ter}} \in [0, T]$ is embedded in the amplitudes. Then, they discretize the BSPDE using FDM, solve it not to $t = 0$ but $t = t_{\mathrm{ter}}$ with quantum ODE solver, and obtain the state $|\boldsymbol{V}(t_{\mathrm{ter}})\rangle$. The inner product of these quantum states, which can be obtained by QAE, corresponds to the expected value of the derivative price at t_{ter} by $E\left[V(t_{\mathrm{ter}})\right] \simeq \sum_{i\in\mathcal{G}} p_i(t_{\mathrm{ter}})V_i(t_{\mathrm{ter}}) = \langle \boldsymbol{p}(t_{\mathrm{ter}})|\boldsymbol{V}(t_{\mathrm{ter}})\rangle$. Discounting this expected value by the risk-free interest rate yields the present price of the derivative [1].

Our algorithm is a variational version of Ref. [108]. Instead of using the quantum ODE solver and QAE, we use VQS and the SWAP test, respectively. This enables derivatives pricing by BSPDE to be realized on a small-sized quantum computer.

5.2.2 Derivative pricing

To evaluate the price of a derivative, we need to model the dynamics of the prices of the underlying asset. We adopt the BS model [35], in which the prices of the underlying assets are assumed to follow geometric Brownian motions. That is, we suppose that the prices of d underlying assets at $t \in [0, T]$ are stochastic processes $\boldsymbol{S}(t) = (S_1(t), S_2(t), \ldots, S_d(t))^\top \in \mathbb{R}^d_+$ that, under the risk-neutral measure, obey stochastic differential equations

$$dS_i(t) = rS_i(t)dt + \sigma_i S_i(t)dW_i(t). \tag{5.1}$$

Here, $r > 0$ is the risk-free interest rate, $\sigma_i > 0$ are volatility of the underlying assets, and they satisfy $0 < r < \frac{\sigma_i^2}{2}$ for all $i \in [d]$. $W_i(t)$ are Brownian motions that satisfy $dW_i dW_j = \rho_{ij}dt, (i, j) \in [d] \times [d]$ with the correlation matrix $(\rho_{ij})_{1\leq i,j\leq d}$, which satisfies $\rho_{ii} = 1$ and $-1 < \rho_{ij} = \rho_{ji} < 1$ for $i \neq j$.

Derivatives are characterized by the payoff function f_{pay} at the maturity and the payoff conditions, which must be satisfied in order for the payoff to arise. We describe the typical cases of the payoff functions and the payoff conditions later. The price of the derivative is obtained as the conditional expected value of the payoff, conditioned on the price of the underlying assets, discounted by the risk-free rate [2]. That is, given the underlying asset prices at time t as $\boldsymbol{s} = (s_1, \ldots, s_d)^\top \in \mathbb{R}^d_+$, and the payoff function at

maturity T as $f_{\text{pay}}(\boldsymbol{S}(T))$, the price of the derivative is

$$V(t, \boldsymbol{s}) = E_Q \left[e^{-r(T-t)} f_{\text{pay}}(\boldsymbol{S}(T)) \mathbb{1}_{\text{NB}} \middle| \boldsymbol{S}(t) = \boldsymbol{s} \right], \tag{5.2}$$

where E_Q is the expected value under the so-called risk-neutral measure. Note that $\boldsymbol{S}(T)$ is a vector of random variable resulting from the time evolution of Eq. (5.1) from t to T with the condition $\boldsymbol{S}(t) = \boldsymbol{s}$. $\mathbb{1}_{\text{NB}}$ is a random variable that takes 1 if the payoff conditions are satisfied or 0 otherwise.

The goal of derivative pricing is to find the present price of the derivative, that is, $V(0, \boldsymbol{s}_0)$, where $\boldsymbol{s}_0 = (s_{1,0}, \ldots, s_{d,0})^\top \in \mathbb{R}_+^d$ is the present price of the underlying assets. To this end, we use the BSPDE, which describes the time evolution of $V(t, \boldsymbol{s})$ [2]. That is, the derivative price $V(t, \boldsymbol{s})$ is the solution of the BSPDE

$$\begin{aligned} &\frac{\partial}{\partial t} V(t, \boldsymbol{s}) + \frac{1}{2} \sum_{i,j=1}^{d} \sigma_i \sigma_j s_i s_j \rho_{ij} \frac{\partial^2}{\partial s_i \partial s_j} V(t, \boldsymbol{s}) \\ &+ r \left(\sum_{i=1}^{d} s_i \frac{\partial}{\partial s_i} V(t, \boldsymbol{s}) - V(t, \boldsymbol{s}) \right) = 0 \end{aligned} \tag{5.3}$$

on $[0, T) \times D$ with the boundary conditions

$$V(T, \boldsymbol{s}) = f_{\text{pay}}(\boldsymbol{s}), \tag{5.4}$$

$$\begin{aligned} &V(t, (s_1, \ldots, s_{i-1}, u_i, s_{i+1}, \ldots, s_d)^\top) \\ &=: V_i^{\text{UB}}(t, \boldsymbol{s}_{\wedge i}), \text{ for } i \in [d], \end{aligned} \tag{5.5}$$

$$\begin{aligned} &V(t, (s_1, \ldots, s_{i-1}, l_i, s_{i+1}, \ldots, s_d)^\top) \\ &=: V_i^{\text{LB}}(t, \boldsymbol{s}_{\wedge i}), \text{ for } i \in [d]. \end{aligned} \tag{5.6}$$

where u_i, l_i are upper and lower bounds of the i-th asset price respectively, and $D := (l_1, u_1) \times \cdots \times (l_d, u_d)$. $V_i^{\text{UB}}, V_i^{\text{LB}}$ are upper and lower boundary conditions for the i-th asset. The boundary conditions in some typical cases of the payoff function and the payoff condition are as follows.

1. If an *up and out barrier* is set on the i-th asset, the payoff is zero if the asset price $S_i(t)$ exceeds u_i at least once before maturity, and then the boundary condition is

$$V_i^{\text{UB}}(t, \boldsymbol{s}_{\wedge i}) = 0. \tag{5.7}$$

   Similarly, if an *down and out barrier* is set on i-th asset, the payoff is zero if the asset price falls below l_i at least once before maturity, and then, the boundary condition is

$$V_i^{\text{LB}}(t, \boldsymbol{s}_{\wedge i}) = 0. \tag{5.8}$$

2. Suppose that the payoff at maturity T is given by

$$f_{\text{pay}}(\boldsymbol{S}(T)) = \max(a_0 + \sum_{i=1}^{d} a_i S_i(T), 0), \tag{5.9}$$

with $a_0, \dots, a_d \in \mathbb{R}$. This is the case with many derivatives. In this form of payoff function, upper boundary or lower boundary can be set depending on the values of $a_0, \dots, a_d$. In some cases, if either of $\{S_i(t)\}_{i\in[d]}$ is sufficiently high or low at some time $t \in (0, T)$, the payoff at T is highly likely to be positive. For example, in the case of the basket call option, that is, $a_0 < 0, a_1, \dots, a_d > 0$, if $\boldsymbol{S}(t) = \boldsymbol{s}$ such that $s_i \gg -a_0/a_i$ for some $i \in [d]$, $f_{\text{pay}}(\boldsymbol{S}(T))$ is likely to be positive. In this situation, the derivative price is approximately equal to $E_Q\left[e^{-r(T-t)}\left(a_0 + \sum_{i=1}^{d} a_i S_i(T)\right) | \boldsymbol{S}(t) = \boldsymbol{s}\right] = e^{-r(T-t)}a_0 + \sum_{i=1}^{d} a_i s_i$. Thus, we can set

$$V_i^{\text{UB}}(t, \boldsymbol{s}_{\wedge i}) = e^{-r(T-t)}a_0 + \sum_{1\le j\le d, j\neq i} a_j s_j + a_i u_i, \tag{5.10}$$

for sufficiently large u_i. In some other cases, e.g. when $a_i < 0$ and $a_j > 0$ for $j \neq i$, we can set

$$V_i^{\text{LB}}(t, \boldsymbol{s}_{\wedge i}) = e^{-r(T-t)}a_0 + \sum_{1\le j\le d, j\neq i} a_j s_j + a_i l_i, \tag{5.11}$$

for sufficiently small l_i.

5.2.3 Finite difference method for the BSPDE

Consider solving Eq. (5.3) using the FDM. In the FDM, we discretize the PDE with respect to the underlying asset prices and obtain the ODE. Then, we can use a numerical solver for ODEs, such as the Euler method, Runge-Kutta method, etc [110]. Note that the BSPDE is often simplified by log-transforming the asset prices as in [108]. However, it is more convenient not to perform a log-transformation to solve the BSPDE by VQS. This is because our formulation presented in Sec. 5.3 can only handle linear boundary conditions with respect to s_i as shown in Appendix A3.2, but a logarithmic transformation will result in the terms like e^{s_i}. Thus, we do not perform the log-transformation in this work.

First, the value range of each underlying asset price s_i is split into n_{gr}

grids. That is, we take

$$\boldsymbol{x}^{(k)} = \left(x_1^{(k_1)}, \dots, x_d^{(k_d)}\right)^\top, \tag{5.12}$$

$$x_i^{(k_i)} \coloneqq l_i + (k_i + 1)\, h_i, \tag{5.13}$$

$$k = \sum_{i=1}^{d} n_{\mathrm{gr}}^{d-i} k_i + 1, \tag{5.14}$$

$$k_i \coloneqq 0, \dots, n_{\mathrm{gr}} - 1, \tag{5.15}$$

$$h_i \coloneqq \frac{u_i - l_i}{n_{\mathrm{gr}} + 1}. \tag{5.16}$$

for $i \in [d]$. By this discretization, we approximate $V(t, \boldsymbol{s})$ by a vector

$$\boldsymbol{V}(t) \coloneqq \left(V(t, \boldsymbol{x}^{(1)}), V(t, \boldsymbol{x}^{(2)}), \dots, V(t, \boldsymbol{x}^{(N_{\mathrm{gr}})})\right)^\top, \tag{5.17}$$

where $N_{\mathrm{gr}} = n_{\mathrm{gr}}^d$. We also replace the differentials by differences as,

$$\frac{\partial V(t, \boldsymbol{x}^{(k)})}{\partial s_i} \to \frac{V(t, \boldsymbol{x}^{(k)} + h_i \boldsymbol{e}_i) - V(t, \boldsymbol{x}^{(k)} - h_i \boldsymbol{e}_i)}{2h_i}, \tag{5.18}$$

$$\begin{aligned}\frac{\partial^2 V(t, \boldsymbol{x}^{(k)})}{\partial s_i^2} \to \frac{1}{h_i^2} \Big(& V(t, \boldsymbol{x}^{(k)} + h_i \boldsymbol{e}_i) + V(t, \boldsymbol{x}^{(k)} - h_i \boldsymbol{e}_i) \\ & - V(t, \boldsymbol{x}^{(k)}) \Big)\end{aligned} \tag{5.19}$$

$$\begin{aligned}\frac{\partial^2 V(t, \boldsymbol{x}^{(k)})}{\partial s_i \partial s_j} \to \frac{1}{4h_i h_j} \Big(& V(t, \boldsymbol{x}^{(k)} + h_i \boldsymbol{e}_i + h_j \boldsymbol{e}_j) \\ & + V(t, \boldsymbol{x}^{(k)} - h_i \boldsymbol{e}_i - h_j \boldsymbol{e}_j) \\ & - V(t, \boldsymbol{x}^{(k)} - h_i \boldsymbol{e}_i + h_j \boldsymbol{e}_j) \\ & - V(t, \boldsymbol{x}^{(k)} - h_i \boldsymbol{e}_i + h_j \boldsymbol{e}_j) \Big),\end{aligned} \tag{5.20}$$

where $\boldsymbol{e}_i = (\underbrace{0, \dots, 0}_{i-1}, 1, \underbrace{0, \dots, 0}_{d-i})^\top, i \in [d]$ is a unit vector of the i-direction.
Introducing $\bar{V}(\tau, \boldsymbol{x}^{(i)}) \coloneqq V(T - t, \boldsymbol{x}^{(i)}), i \in [d]$ and $\bar{\boldsymbol{V}}(\tau) \coloneqq \left(\bar{V}(\tau, \boldsymbol{x}^{(1)}), \bar{V}(\tau, \boldsymbol{x}^{(2)}), \dots, \bar{V}(\tau, \boldsymbol{x}^{(N_{\mathrm{gr}})})\right)^\top$, we eventually obtain the ODE

$$\frac{d}{d\tau} \bar{\boldsymbol{V}}(\tau) = F \bar{\boldsymbol{V}}(\tau) + \boldsymbol{C}(\tau) \tag{5.21}$$

and initial condition

$$\bar{\boldsymbol{V}}(0) = \left(f_{\mathrm{pay}}(\boldsymbol{x}^{(1)}), \dots, f_{\mathrm{pay}}(\boldsymbol{x}^{(N_{\mathrm{gr}})})\right)^\top. \tag{5.22}$$

Here, F is an $N_{\rm gr} \times N_{\rm gr}$ real matrix,

$$F := F^{\rm 1st} + F^{\rm 2nd} - rI^{\otimes d} \tag{5.23}$$

$$\begin{aligned} F^{\rm 2nd} :=& \sum_{i=1}^{d} \frac{\sigma_i^2}{2h_i^2} I^{\otimes i-1} \otimes D_{x_i}^{\rm 2nd} \otimes I^{\otimes d-i} \\ &+ \sum_{i=1}^{d-1} \sum_{j=i+1}^{d} \frac{\sigma_i \sigma_j \rho_{ij}}{4h_i h_j} \\ &\times I^{\otimes i-1} \otimes D_{x_i}^{\rm 1st} \otimes I^{\otimes j-i-1} \otimes D_{x_j}^{\rm 1st} \otimes I^{\otimes d-j} \end{aligned} \tag{5.24}$$

$$F^{\rm 1st} := r \sum_{i=1}^{d} \frac{1}{2h_i} I^{\otimes i-1} \otimes D_{x_i}^{\rm 1st} \otimes I^{\otimes d-i}, \tag{5.25}$$

where I is a $n_{\rm gr} \times n_{\rm gr}$ identity matrix, $D_{x_i}^{\rm 1st}, D_{x_i}^{\rm 2nd}$ are $n_{\rm gr} \times n_{\rm gr}$ real matrices. $\boldsymbol{C}(\tau)$ is a vector corresponding to the boundary conditions. The elements of the $D_{x_i}^{\rm 1st}, D_{x_i}^{\rm 2nd}$, and $\boldsymbol{C}(\tau)$ are shown in Appendix A3.1. $n_{\rm gr}$ has to be proportional to $O(\epsilon^{-1/2})$ to obtain the present price of the derivative within the accuracy ϵ [108]. Then, the dimension of $\bar{\boldsymbol{V}}(\tau)$ is $O((1/\epsilon)^{d/2})$. Thus, it becomes difficult to solve the BSPDE discretized by FDM using the classical algorithm when multiple assets need to be considered.

5.3 Proposed method

In this section, we describe the variational quantum algorithm for derivative pricing and the computational complexity of the proposed method. The overall algorithm is shown in Algo. 3. We assume that $n_{\rm gr}^d = 2^n$ with the n-qubit system.

Algorithm 3 Derivative Pricing with Variational Quantum Algorithms

1: Prepare $a_p U_p$ such that $a_p U_p |0\rangle = |\psi_p\rangle \simeq \sum_{k=1}^{N_{\rm gr}} p_k(t_{\rm ter}) |k\rangle$ by VQS for Fokker-Planck equation or quantum generative models.
2: Prepare $\alpha_V U_V$ such that $\alpha_V U_V |0\rangle = |\psi_V\rangle \simeq \sum_{k=1}^{N_{\rm gr}} f_{\rm pay}(\boldsymbol{x}^{(k)}) |k\rangle$ by quantum generative models.
3: Calculate $|\tilde{v}(\boldsymbol{\theta}(\tau_{\rm ter}))\rangle$ by performing VQS from $\tau = 0$ to $\tau = \tau_{\rm ter}$.
4: Perform the SWAP test and get an estimation of $|\langle \psi^p | \tilde{v}(\boldsymbol{\theta}(\tau_{\rm ter}))\rangle|^2$
5: $V_0 \leftarrow e^{-rt_{\rm ter}} \langle \psi_p | \tilde{v}(\boldsymbol{\theta}(\tau_{\rm ter}))\rangle$.

First, we set $\tau_{\rm ter} = T - t_{\rm ter}$, where $t_{\rm ter}$ is defined in Eq. (4.14). We also set N_τ, which is the number of steps for VQS. To perform VQS, we need to represent the operator corresponding to F in Eq. (5.23) and the operator $\tilde{G}$ such that $\tilde{G} |0\rangle = |\boldsymbol{C}\rangle = \sum_{k=1}^{N_{\rm gr}} C_k(\tau) |k\rangle$ by a linear combination of quantum gates, respectively, because of the assumptions Eqs. (2.51) and (2.52). Such

decomposition can be obtained in a similar way to Ref. [70, 30] and is shown in Appendix A3.2. F can be represented as a sum of $O(d^2n^4)$ unitaries each of which requires at most $O(n^2)$ gates to be implemented. $\tilde{G}$ for typical boundary conditions discussed in Sec. 5.2.3 can be represented by $O(d^3n^2)$ unitaries, which require at most $O(n^2)$ gates to be implemented.

Second, we prepare the unnormalized state

$$\begin{aligned} |\psi_p\rangle &\coloneqq \alpha_p U_p |0\rangle \\ &\simeq |\boldsymbol{p}(t_{\mathrm{ter}})\rangle \\ &= \sum_{k=1}^{N_{\mathrm{gr}}} p_k(t_{\mathrm{ter}}) |k\rangle , \end{aligned} \tag{5.26}$$

where $\alpha_p \in \mathbb{C}$, and U_p is an quantum gate. $p_k(t_{\mathrm{ter}})$ is a probability that the underlying asset prices is on $\boldsymbol{x}^{(k)}$ at t_{ter}. We can obtain such α_p and U_p by solving the Fokker-Planck equation, which describes the time evolution of the probability density function, using VQS [70, 30]. Alternatively, they can also be obtained by quantum generative models [15, 14, 12, 13, 17] since the probability density function of the underlying asset price at any $t \in [0, T]$ can be obtained analytically under the BS model (see Eq. (5.41) in Sec. 5.3.1).

Third, we prepare $\alpha_V \in \mathbb{C}$ and U_V such that $\alpha_V U_V |0\rangle =: |\psi_V\rangle$ approximates the initial state of the discretized BSPDE, that is,

$$\begin{aligned} |\psi_V\rangle &\simeq \left|\bar{\boldsymbol{V}}(0)\right\rangle \\ &= \sum_{k=1}^{N_{\mathrm{gr}}} f_{\mathrm{pay}}(\boldsymbol{x}^{(k)}) |k\rangle \end{aligned} \tag{5.27}$$

To find such α_V and U_V, we can use the quantum generative models [15, 14, 12, 13, 17].

Fourth, we solve the BSPDE from $\tau = 0$ to τ_{ter} using VQS and obtain an unnormalized state

$$|\tilde{v}(\boldsymbol{\theta}(\tau_{\mathrm{ter}}))\rangle \simeq \left|\bar{\boldsymbol{V}}(\tau_{\mathrm{ter}})\right\rangle = \sum_{k=1}^{N_{\mathrm{gr}}} \bar{\boldsymbol{V}}_k(\tau_{\mathrm{ter}}, \boldsymbol{x}^{(k)}) |k\rangle \tag{5.28}$$

where

$$\begin{aligned} |\tilde{v}(\boldsymbol{\theta}(\tau_{\mathrm{ter}}))\rangle \coloneqq{}& \theta_0(\tau_{\mathrm{ter}}) R_1(\theta_1(\tau_{\mathrm{ter}})) R_2(\theta_2(\tau_{\mathrm{ter}})) \\ & \cdots R_{N_a}(\theta_{N_a}(\tau_{\mathrm{ter}})) |\psi_V\rangle \end{aligned} \tag{5.29}$$

$\{R_k\}_{k\in[N_a]}$ are parameterized quantum circuits, and $\boldsymbol{\theta}(\tau_{\mathrm{ter}}) \coloneqq (\theta_0(\tau_{\mathrm{ter}}), \ldots, \theta_{N_a}(\tau_{\mathrm{ter}}))^\top \in \mathbb{R}^{N_a+1}$ is the variational parameters. Note that $\theta_0(0) R_1(\theta_1(0)) R_2(\theta_2(0)) R_{N_a}(\theta_{N_a}(0))$ should be an identity operator to satisfy $|\tilde{v}(\boldsymbol{\theta}(0))\rangle \simeq \left|\bar{\boldsymbol{V}}(0)\right\rangle$. For example, the ansatz shown in Fig. 5.1 in Sec. 5.4 with even number of layers

can be used as $\{R_k\}_{k\in[N_a]}$ that satisfies this condition with the parameters $\boldsymbol{\theta}(0) = (0,\dots,0)^\top$ since RY gates are identity for the parameters, and CZ layers cancel each other and also become identity.

Finally, we use the SWAP test [109] for two normalized states $U_p\,|0\rangle$, $R_1(\theta_1(\tau_{\rm ter}))\cdots R_{N_a}(\theta_{N_a}(\tau_{\rm ter}))U_V\,|0\rangle$ and obtain

$$\begin{aligned}&|\langle\psi_p|\tilde{v}(\boldsymbol{\theta}(\tau_{\rm ter}))\rangle|^2\\&= |\alpha_p\alpha_V\theta_0(\tau_{\rm ter})|^2\\&\times\left|\langle 0|\,U_p^\dagger R_1(\theta_1(\tau_{\rm ter}))\cdots R_{N_a}(\theta_{N_a}(\tau_{\rm ter}))U_V\,|0\rangle\right|^2.\end{aligned}\tag{5.30}$$

The present price of the derivative is approximated by the inner product $\left\langle \boldsymbol{p}(t_{\rm ter})\middle|\bar{\boldsymbol{V}}(\tau_{\rm ter})\right\rangle$ discounted by the risk-free rate. We can approximate the inner product by the square root of the result of the SWAP test and obtain the present price of the derivative by

$$V_0 \simeq e^{-rt_{\rm ter}}\,\langle\psi_p|\tilde{v}(\boldsymbol{\theta}(\tau_{\rm ter}))\rangle\,.\tag{5.31}$$

For the third and fourth parts, we may take a slightly different approach. That is, we find $\boldsymbol{\theta}(0)$ such that

$$\left|\bar{\boldsymbol{V}}(0)\right\rangle \simeq \theta_0(0)R_1(\theta_1(0))R_2(\theta_2(0))\cdots R_{N_a}(\theta_{N_a}(0))\,|0\rangle\tag{5.32}$$

and obtain

$$\begin{aligned}|\tilde{v}(\boldsymbol{\theta}(\tau_{\rm ter}))\rangle &= \theta_0(\tau_{\rm ter})R_1(\theta_1(\tau_{\rm ter}))R_2(\theta_2(\tau_{\rm ter}))\\&\quad\cdots R_{N_a}(\theta_{N_a}(\tau_{\rm ter}))\,|0\rangle\\&\simeq \left|\bar{\boldsymbol{V}}(\tau_{\rm ter})\right\rangle\end{aligned}\tag{5.33}$$

using VQS. This approach may reduce the number of gates by eliminating U_V, but since the ansatz for the initial state also serves as the ansatz for VQS, the number of gates required for the ansatz may become larger. For this reason, it is difficult to say which approach is better in general, but we adopt the one in Algo. 3 for the numerical simulation in Sec. 5.4.

5.3.1 The number of measurements in the SWAP test

In this subsection, we estimate the number of measurements required for the SWAP test. For simplicity, we consider the case where $|\tilde{v}(\boldsymbol{\theta}(\tau_{\rm ter}))\rangle = \left|\bar{\boldsymbol{V}}(\tau_{\rm ter})\right\rangle$ and $|\psi_p\rangle = |\boldsymbol{p}(t_{\rm ter})\rangle$. We perform the SWAP test for two normalized states $|\tilde{\boldsymbol{p}}\rangle$ and $\left|\tilde{\bar{\boldsymbol{V}}}\right\rangle$ such that $|\boldsymbol{p}(t_{\rm ter})\rangle = \alpha\,|\tilde{\boldsymbol{p}}\rangle\,, \left|\bar{\boldsymbol{V}}(\tau_{\rm ter})\right\rangle = \beta\left|\tilde{\bar{\boldsymbol{V}}}\right\rangle$,

where

$$\alpha = \sqrt{\sum_{k=1}^{N_{\mathrm{gr}}} p_k(t_{\mathrm{ter}})^2}, \tag{5.34}$$

$$\beta = \sqrt{\sum_{k=1}^{N_{\mathrm{gr}}} \bar{V}(\tau_{\mathrm{ter}}, \boldsymbol{x}^{(k)})^2}. \tag{5.35}$$

To obtain the value of the inner product $\left|\left\langle \tilde{\boldsymbol{p}} \middle| \tilde{\bar{\boldsymbol{V}}} \right\rangle\right|^2$ with precision $\bar{\varepsilon}$, the SWAP test requires $O(\frac{1}{\bar{\varepsilon}^2})$ measurements [109]. When we have the estimation $\widetilde{\left|\left\langle \tilde{\boldsymbol{p}} \middle| \tilde{\bar{\boldsymbol{V}}} \right\rangle\right|^2}$ such that

$$\left| \left|\left\langle \tilde{\boldsymbol{p}} \middle| \tilde{\bar{\boldsymbol{V}}} \right\rangle\right|^2 - \widetilde{\left|\left\langle \tilde{\boldsymbol{p}} \middle| \tilde{\bar{\boldsymbol{V}}} \right\rangle\right|^2} \right| < \bar{\varepsilon}, \tag{5.36}$$

the estimation of the inner product of unnormalized states $\widetilde{\left|\left\langle \boldsymbol{p}(t_{\mathrm{ter}}) \middle| \bar{\boldsymbol{V}}(\tau_{\mathrm{ter}}) \right\rangle\right|^2}$ satisfies

$$\left| \left|\left\langle \boldsymbol{p}(t_{\mathrm{ter}}) \middle| \bar{\boldsymbol{V}}(\tau_{\mathrm{ter}}) \right\rangle\right|^2 - \widetilde{\left|\left\langle \boldsymbol{p}(t_{\mathrm{ter}}) \middle| \bar{\boldsymbol{V}}(\tau_{\mathrm{ter}}) \right\rangle\right|^2} \right| < \alpha^2\beta^2\bar{\varepsilon}. \tag{5.37}$$

Thus, $O(\frac{\alpha^4\beta^4}{\varepsilon^2})$ measurements are required to obtain $\left|\left\langle \boldsymbol{p}(t_{\mathrm{ter}}) \middle| \bar{\boldsymbol{V}}(\tau_{\mathrm{ter}}) \right\rangle\right|^2$ with precision $\varepsilon \coloneqq \alpha^2\beta^2\bar{\varepsilon}$. Note that since we can classically calculate α by the analytical form of $p(t, \boldsymbol{s})$, and β is calculated by $\alpha_V\theta_0(\tau_{\mathrm{ter}})$, we can determine the number of measurements before the SWAP test from VQS results.

To estimate the number of measurements of the SWAP test, we estimate $\alpha^2\beta^2$, which is calculated as

$$\begin{aligned}
\alpha^2\beta^2 &= \left(\sum_{k=1}^{N_{\mathrm{gr}}} p_k(t_{\mathrm{ter}})^2\right)\left(\sum_{k=1}^{N_{\mathrm{gr}}} \bar{V}(\tau_{\mathrm{ter}}, \boldsymbol{x}^{(k)})^2\right) \\
&= \left(\sum_{k=1}^{N_{\mathrm{gr}}} p_k(t_{\mathrm{ter}})^2\right)\left(\sum_{k=1}^{N_{\mathrm{gr}}} f_{\mathrm{pay}}(\boldsymbol{x}^{(k)})^2\right) \\
&\quad \times \frac{\sum_{k=1}^{N_{\mathrm{gr}}} \bar{V}(\tau_{\mathrm{ter}}, \boldsymbol{x}^{(k)})^2}{\sum_{k=1}^{N_{\mathrm{gr}}} f_{\mathrm{pay}}(\boldsymbol{x}^{(k)})^2}.
\end{aligned} \tag{5.38}$$

Although it is difficult to estimate the factor $\sum_{k=1}^{N_{\mathrm{gr}}} \bar{V}(\tau_{\mathrm{ter}}, \boldsymbol{x}^{(k)})^2 / \sum_{k=1}^{N_{\mathrm{gr}}} f_{\mathrm{pay}}(\boldsymbol{x}^{(k)})^2$ in advance, we assume that the factor is bounded by some constant ζ. This assumption means that the rate of change in derivative prices over time is suppressed by a certain constant. Under the assumption, we estimate

$\left(\sum_{k=1}^{N_{\mathrm{gr}}} f_{\mathrm{pay}}(\boldsymbol{x}^{(k)})^2\right)\left(\sum_{k=1}^{N_{\mathrm{gr}}} p_k(t_{\mathrm{ter}})^2\right)$. We assume that $f_{\mathrm{pay}}(\boldsymbol{x})$ for $\boldsymbol{x} \in D$ is upper bounded by some constant B. For example, in the case of the basket call option,

$$\begin{aligned} f_{\mathrm{pay}}(\boldsymbol{x}) &= \max(a_0 + \sum_{i=1}^{d} a_i x_i - K, 0) \\ &\leq a_0 + \sum_{i=1}^{d} a_i x_i \\ &\leq a_0 + \sum_{i=1}^{d} a_i u_i \end{aligned} \tag{5.39}$$

holds. From this assumption, we obtain

$$\begin{aligned} \sum_{k=1}^{N_{\mathrm{gr}}} f_{\mathrm{pay}}(\boldsymbol{x}^{(k)})^2 &\leq \sum_{k=1}^{N_{\mathrm{gr}}} B^2 \\ &= N_{\mathrm{gr}} B^2. \end{aligned} \tag{5.40}$$

On the other hand, the probability density function of d-dimensional geometric Brownian motion with $\boldsymbol{x}(0) = \boldsymbol{x}_0 \coloneqq (x_{0,1}, \ldots, x_{0,d})^\top$ is

$$\begin{aligned} p(t, \boldsymbol{x}) = &\frac{1}{(2\pi t)^{d/2}\left(\prod_{i=1}^{d} \sigma_i x_i\right)\sqrt{\det \rho}} \\ &\times \exp\left(-\frac{1}{2}(\ln \boldsymbol{x} - \boldsymbol{\mu})^\top \Sigma^{-1} (\ln \boldsymbol{x} - \boldsymbol{\mu})\right), \end{aligned} \tag{5.41}$$

where

$$\boldsymbol{\mu} = \left(\left(r - \frac{\sigma_1^2}{2}\right)t - x_{0,1}, \ldots, \left(r - \frac{\sigma_d^2}{2}\right)t - x_{0,d}\right)^\top. \tag{5.42}$$

The square of probability density function is

$$\begin{aligned} p(t, \boldsymbol{x})^2 = &\left(\frac{1}{(2\pi t)^{d/2}\left(\prod_{i=1}^{d} \sigma_i x_i\right)\sqrt{\det \rho}}\right)^2 \\ &\times \exp\left(-(\ln \boldsymbol{x} - \boldsymbol{\mu})^\top \Sigma^{-1} (\ln \boldsymbol{x} - \boldsymbol{\mu})\right) \\ = &\frac{\gamma(t)}{\prod_{i=1}^{d} x_i} \frac{1}{(2\pi t)^{d/2}\left(\prod_{i=1}^{d} \frac{\sigma_i x_i}{2}\right)\sqrt{\det \rho}} \\ &\times \exp\left(-\frac{1}{2}(\ln \boldsymbol{x} - \boldsymbol{\mu})^\top \left(\frac{1}{2}\Sigma\right)^{-1} (\ln \boldsymbol{x} - \boldsymbol{\mu})\right) \\ = &\frac{\gamma(t)}{\prod_{i=1}^{d} x_i} \varphi(t, \boldsymbol{x}), \end{aligned} \tag{5.43}$$

where

$$\gamma(t) = \frac{1}{(8\pi t)^{d/2} \prod_{i=1}^{d} \sigma_i}, \tag{5.44}$$

and $\varphi(t, \boldsymbol{x})$ is a probability density function of some log-normal distribution. Using the probability distribution function, the square sum of the discretized density function is represented by

$$\begin{aligned}
\sum_{k=1}^{N_{\mathrm{gr}}} p_k(t_{\mathrm{ter}})^2 &= \sum_{k=1}^{N_{\mathrm{gr}}} \left(p(t_{\mathrm{ter}}, \boldsymbol{x}^{(k)}) \right)^2 \left(\prod_{i=1}^{d} h_i \right)^2 \\
&= \sum_{k=1}^{N_{\mathrm{gr}}} \frac{\gamma(t_{\mathrm{ter}})}{\prod_{i=1}^{d} x_i^{(k_i)}} \varphi(t_{\mathrm{ter}}, \boldsymbol{x}^{(k)}) \left(\prod_{i=1}^{d} h_i \right)^2 \\
&\leq \frac{\gamma(t_{\mathrm{ter}})}{\prod_{i=1}^{d} l_i} \sum_{k=1}^{N_{\mathrm{gr}}} \varphi(t_{\mathrm{ter}}, \boldsymbol{x}^{(k)}) \left(\prod_{i=1}^{d} h_i \right)^2 \\
&\simeq \frac{\gamma(t_{\mathrm{ter}})}{\prod_{i=1}^{d} l_i} \prod_{i=1}^{d} h_i \int_{\mathbb{R}_+^d} \varphi(t_{\mathrm{ter}}, \boldsymbol{x}) d\boldsymbol{x} \\
&= \frac{\gamma(t_{\mathrm{ter}})}{\prod_{i=1}^{d} l_i} \prod_{i=1}^{d} h_i \\
&= \frac{\gamma(t_{\mathrm{ter}})}{\prod_{i=1}^{d} l_i} \frac{1}{N_{\mathrm{gr}}} \prod_{i=1}^{d} (u_i - l_i).
\end{aligned} \tag{5.45}$$

From Eqs. (5.40) and (5.45), we obtain

$$\begin{aligned}
\alpha^2 \beta^2 &\lesssim \zeta B^2 \frac{1}{(8\pi t_{\mathrm{ter}})^{d/2}} \prod_{i=1}^{d} \frac{1}{\sigma_i} \left(\frac{u_i}{l_i} - 1 \right) \\
&=: \Xi.
\end{aligned} \tag{5.46}$$

Since t_{ter} is lower bounded by

$$\begin{aligned}
t_{\mathrm{ter}} &= \min \left\{ \frac{2 \left(\log \left(\frac{u_1}{s_{1,0}} \right) \right)^2}{25\sigma_1^2 \log \left(\frac{2\tilde{A}d(d+1)}{\epsilon} \right)}, \ldots, \frac{2 \left(\log \left(\frac{u_d}{s_{d,0}} \right) \right)^2}{25\sigma_d^2 \log \left(\frac{2\tilde{A}d(d+1)}{\epsilon} \right)} \right. \\
&\quad \left. \frac{2 \left(\log \left(\frac{s_{1,0}}{l_1} \right) \right)^2}{25\sigma_1^2 \log \left(\frac{2\tilde{A}d(d+1)}{\epsilon} \right)}, \ldots, \frac{2 \left(\log \left(\frac{s_{d,0}}{l_d} \right) \right)^2}{25\sigma_d^2 \log \left(\frac{2\tilde{A}d(d+1)}{\epsilon} \right)} \right\}, \\
&\geq \frac{2 \left(\log \chi_{\min} \right)^2}{25\sigma_{\max}^2} \left(\log \frac{2\tilde{A}d(d+1)}{\epsilon} \right)^{-1},
\end{aligned} \tag{5.47}$$

where $\sigma_{\max} \coloneqq \max_{i\in[d]}\{\sigma_i\}$, and $\chi_{\min} \coloneqq \min_{i\in[d]}\{u_i/s_{i,0}\} \cup \{s_{i,0}/l_i\}$, we obtain

$$\Xi \leq \zeta B^2 \times \left(\frac{5}{4\pi^2}\frac{\xi_{\max}-1}{\log\chi_{\min}}\frac{\sigma_{\max}}{\sigma_{\min}}\right)^d \left(\log\frac{2\tilde{A}d(d+1)}{\epsilon}\right)^{d/2}, \tag{5.48}$$

where $\sigma_{\min} \coloneqq \min_{i\in[d]}\{\sigma_i\}$, and $\xi_{\max} \coloneqq \max_{i\in[d]}\{u_i/l_i\}$[1]. We find that the number of measurements required by the SWAP test is

$$N_{\mathrm{SWAP}} = \frac{\zeta^2 B^4}{\varepsilon^2}\left(\frac{5}{4\pi^2}\frac{\xi_{\max}-1}{\log\chi_{\min}}\frac{\sigma_{\min}}{\sigma_{\max}}\right)^{2d}\left(\log\frac{2\tilde{A}d(d+1)}{\epsilon}\right)^d. \tag{5.49}$$

Note that N_{SWAP} does not have the dependency of the form like $(1/\epsilon)^{O(d)}$, which means that the proposed method achieves a significant speedup over classical FDM with respect to ϵ and d, when the other parts of the proposed method are sufficiently efficient.

Here, we consider the limit of $t_{\mathrm{ter}} \to 0$. This corresponds to retrieving one amplitude of the computational basis from $|\boldsymbol{V}(0)\rangle$ as in [27, 29]. In this case, the probability density function (Eq. (5.41)) is a delta function, which means that the present price of the underlying assets is $\boldsymbol{x}_0$ with probability 1. Assuming that $\boldsymbol{x}_0$ is on a grid point with the index k_0, $p_k(0)$ is 1 for $k = k_0$ and 0 otherwise, and the sum of the squares of $p_k(0)$ is 1. As a result, $\alpha^2\beta^2$ is upper-bounded as follows,

$$\alpha^2\beta^2 \leq \zeta N_{\mathrm{gr}} B^2 \tag{5.50}$$

Thus, the number of the measurement is proportional to $N_{\mathrm{gr}} = n_{\mathrm{gr}}^d$, and the quantum speedup will be lost.

5.3.2 Computational complexity of proposed method

Here, we discuss the computational complexity of our algorithm. We assume that the number of quantum gates required for preparing $|\boldsymbol{p}(t_{\mathrm{ter}})\rangle$ and $|\bar{\boldsymbol{V}}(0)\rangle$ are N^p_{gate} and N^V_{gate} respectively. We also assume that the number of measurements required to prepare $|\boldsymbol{p}(t_{\mathrm{ter}})\rangle$ and $|\bar{\boldsymbol{V}}(0)\rangle$ are N^p_{measure} and N^V_{measure} respectively. $N^p_{\mathrm{gate}}, N^V_{\mathrm{gate}}, N^p_{\mathrm{measure}}$, and N^V_{measure} depend on the implementation of the generative models, but we assume that all of them are $O(\mathrm{poly}(d\log(1/\epsilon)))$. This means that we assume that the generative models

[1] When $\xi_{\max}$ is close to 1, one may find it strange that as Ξ decreases exponentially with respect to the number of assets d, and then, the number of measurements also decrease exponentially. We show that such an exponential decrease does not occur by evaluating the lower bound of Ξ. See Appendix A3.5 for details.

efficiently generate the (unnormalized) quantum states. Note that VQS requires controlled versions of U_k^L, U_l^u in Eq. (2.56), or those of $\mathcal{R}U_v$ where $\mathcal{R}$ is defined in Eq. (A142) (see Appendix A3.4). Since U_k^L and U_l^u are terms of the linear combination of F and $\tilde{G}$, respectively, they are made by $O(n^2) = O(d\log(1/\epsilon))$ gates. Thus, $O(\mathrm{poly}(d\log(1/\epsilon))$ gates are required for the control unitaries of U_k^L and U_l^u. Assuming $\mathcal{R}U_v$ is made by $O(\mathrm{poly}(d\log(1/\epsilon))$ gates, the controlled-$\mathcal{R}U_v$ gate requires $O(\mathrm{poly}(d\log(1/\epsilon))$ gates. Consequently, quantum circuits containing $O(\mathrm{poly}(d\log(1/\epsilon))$ quantum gates is required for VQS. We assume that the number of measurements to estimate $\mathcal{M}_{i,j}$ and $\mathcal{V}_i$ are $N_{\mathrm{measure}}^{\mathrm{VQS}}$. The number of quantum gates to perform the SWAP test is $O(\mathrm{poly}(d\log(1/\epsilon)))$ since the SWAP test requires $O(n) = O(d\log(1/\epsilon))$ quantum gates in addition to the quantum gates to generate $|\boldsymbol{p}(t_{\mathrm{ter}})\rangle$ and $\left|\bar{\boldsymbol{V}}(\tau)\right\rangle$ [109]. The number of measurements for the SWAP test is N_{SWAP} in Eq. (5.49). The summary of the complexities of the proposed method is shown in Table 5.1.

Part of the algorithm	# of quantum gates	# of measurements
Preparing $\left\|\boldsymbol{p}_{\mathrm{ter}}\right\rangle$	N_{gate}^{p}	N_{measure}^{p}
Preparing $\left\|\boldsymbol{V}_{\mathrm{ter}}\right\rangle$	N_{gate}^{V}	N_{measure}^{V}
VQS	$O(\mathrm{poly}(d\log(1/\epsilon)))$	$N_{\mathrm{measure}}^{\mathrm{VQS}} N_\tau$
SWAP test	$O(\mathrm{poly}(d\log(1/\epsilon)))$	N_{SWAP} in Eq.(5.49)

Table 5.1: The complexities of the proposed method.

Note that, although there remains the exponential dependency with respect to d in N_{SWAP}, the time complexity does not have any factor like $(1/\epsilon)^{O(d)}$, as discussed in Sec. 5.3.1. This is the possible advantage of our method since the complexity of the classical FDM and conventional quantum algorithm have a factor like $(1/\epsilon)^{O(d)}$.

5.4 Numerical Results

In this section, we validate the proposed method using numerical calculations. This experiment focuses on a single asset double knock-out barrier option, which contains both *up and out* and *down and out* conditions. According to [111], the analytical solution for the single asset double barrier

option $\tilde{V}$ with an upper bound u and a lower bound l is

$$\begin{aligned}\tilde{V}(t) = S_0 \sum_{n=-\infty}^{\infty} \Bigg\{ & \left(\frac{u^n}{l^n}\right)^c \left[\mathcal{N}(d_{1n}) - \mathcal{N}(d_{2n})\right] \\ & - \left(\frac{u^{n+1}}{l^n S_0}\right)^c \left[\mathcal{N}(d_{3n}) - \mathcal{N}(d_{4n})\right] \Bigg\} \\ & - K e^{-r\tau} \\ & \times \sum_{n=-\infty}^{\infty} \Bigg\{ \left(\frac{u^n}{l^n}\right)^{c-2} \left[\mathcal{N}(d_{1n} - \sigma\sqrt{\tau}) - \mathcal{N}(d_{2n} - \sigma\sqrt{\tau})\right] \\ & - \left(\frac{u^{n+1}}{l^n S_0}\right)^{c-2} \left[\mathcal{N}(d_{3n} - \sigma\sqrt{\tau}) - \mathcal{N}(d_{4n} - \sigma\sqrt{\tau})\right] \Bigg\}, \end{aligned} \tag{5.51}$$

where

$$d_{1n} = \frac{\ln\left(\frac{S_0}{K}\left(\frac{u}{l}\right)^{2n}\right) - \left(r + \frac{\sigma^2}{2}\right)\tau}{\sigma\sqrt{\tau}}, \tag{5.52}$$

$$d_{2n} = \frac{\ln\left(S_0 \frac{u^{2n-1}}{l^{2n}}\right) - \left(r + \frac{\sigma^2}{2}\right)\tau}{\sigma\sqrt{\tau}}, \tag{5.53}$$

$$d_{3n} = \frac{\ln\left(\frac{u^{2n+2}}{K S_0 l^{2n}}\right) - \left(r + \frac{\sigma^2}{2}\right)\tau}{\sigma\sqrt{\tau}}, \tag{5.54}$$

$$d_{4n} = \frac{\ln\left(S_0 \frac{u^{2n+1}}{l^{2n}}\right) - \left(r + \frac{\sigma^2}{2}\right)\tau}{\sigma\sqrt{\tau}}, \tag{5.55}$$

$$c = \frac{2r}{\sigma} + 1, \tag{5.56}$$

and $\mathcal{N}(\cdot)$ is the cumulative distribution function of the standard normal distribution. We compare the results obtained by the proposed method with the analytical solution. We use the Euler method for the time evolution of the parameter (Eq. (2.54)). The step size for the Euler method is $\Delta\tau = 2.5 \times 10^{-5}$. The parameters are $r = 0.001, \sigma := \sigma_1 = 0.3, T = 1, S_0 = 1, l := l_1 = 0.5, u := u_1 = 2.0, K = 1$. The ansatz of VQS for solving the BS model is shown in Fig. 5.1. This ansatz repeats m parameterized layers consisting of n RY gates and an entanglement layer consisting of CZ gates. The ansatz have $n(m+1)$ parameters. We do not consider noise and statistical errors in the simulation of quantum circuits. In addition, we assume that the initial state $\left|\bar{\boldsymbol{V}}(0)\right\rangle$ and $|\boldsymbol{p}(t)\rangle$ for all $t \in [0, T]$ are given. For the simulation of quantum states, we use NumPy [60].

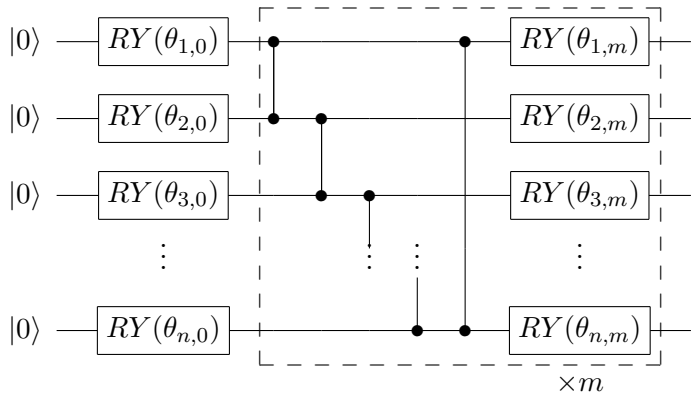


Figure 5.1: In a depth-m circuit, CZ and RY gates (enclosed by dashed lines) are repeated m-times. The circuit has $n(m+1)$ parameters.

5.4.1 Parameter dependencies of VQS results

Before discussing our results, we show the results using the classical FDM in Fig. 5.2. The plotted curves are $V(0, S_0) \simeq e^{-rt} E\left[V(t, S) | S(0) = S_0\right]$ at each $t \in [0, T]$, where $V(t, S)$ is calculated by classical FDM and the expectation is taken with respect to the analytical $p(t, s)$. The error from the analytical solution increases as t increases for $t \geq t_{\text{ter}}$. This is because, in the range greater than t_{ter}, the probability that the underlying asset price exceeds or falls under the boundary conditions is higher. As the number of the grid points increases, the derivative price by FDM gets closer to the analytical solution at t_{ter}. Since $S_0 = 1$ is not on the grid points, the error increases when the probability distribution approaches the indicator function with $t \to 0$.

Fig. 5.3 shows the present price of the derivative calculated by our proposed method. We perform VQS on the simulator and obtain $|\tilde{\boldsymbol{v}}(\boldsymbol{\theta}(\tau))\rangle$, which is an approximation of $\left|\bar{\boldsymbol{V}}(\tau)\right\rangle$. Taking the inner product between $|\tilde{\boldsymbol{v}}(\boldsymbol{\theta}(\tau))\rangle$ and $|\boldsymbol{p}(t)\rangle$, which is calculated by Eqs. (5.26) and (5.41), we obtain the estimation of the present price of the derivative. In the 4 qubits case, the result of VQS is a good approximation to the classical FDM solutions of 16 grid points. The use of the larger number of qubits, i.e., the larger n_{gr}, gives us solutions that are closer to the analytical solution as in the case of the classical FDM. In the case of 6 qubits with 4 layers, the number of parameters is 30, which is smaller than the number of grid points of 64, but the solution is somewhat close to the classical FDM. Due to computational time requirements, we do not run simulations of larger sizes. However, we find that the solution obtained with more layers better approximates the classical FDM solution.

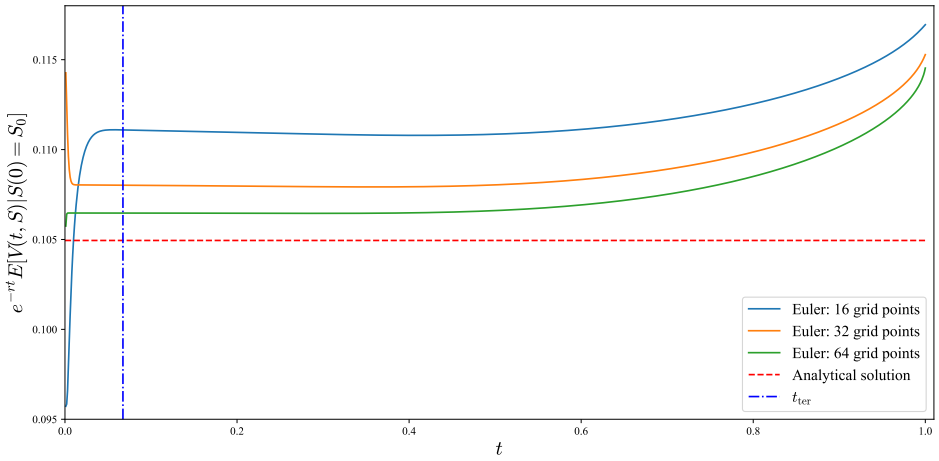


Figure 5.2: The estimated price of the single-asset double barrier option by classical FDM.

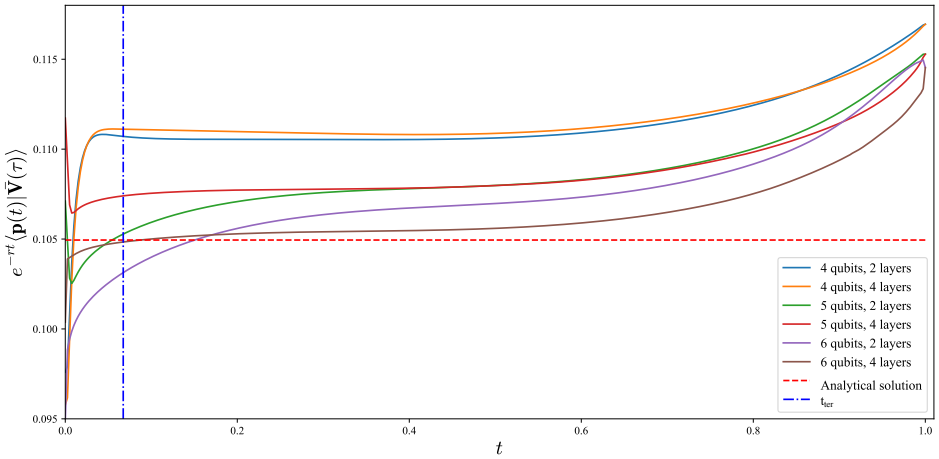


Figure 5.3: The estimated price of the single-asset double barrier option by the proposed method.

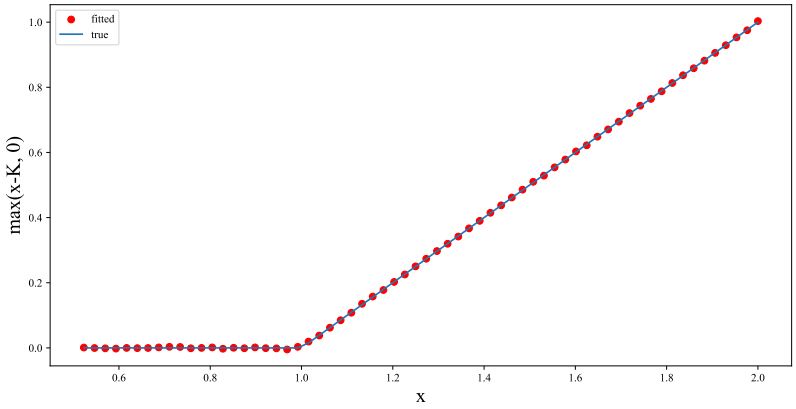


Figure 5.4: Target initial condition for single assets (solid line) and the initial state obtained by fidelity maximization (circle dots). Parameters are $f_{\mathrm{pay}}(x) = \max(x - K, 0), K = 1$. The value corresponding to fidelity satisfies $|1 - \alpha_0^{-2}| \left\langle \bar{\boldsymbol{V}}(0) \middle| u(\boldsymbol{\theta}_0) \right\rangle|^2| \leq 1.2 \times 10^{-5}$.

5.4.2 Possibility of initial state generation

To solve the terminal value problem of the BSPDE, it is necessary to prepare the (unnormalized) initial state $\left|\bar{\boldsymbol{V}}(0)\right\rangle = \sum_k f_{\mathrm{pay}}(\boldsymbol{x}^{(k)}) \left|k\right\rangle$, which we assumed to be given in the previous subsection. Here, we show by simulation that for a typical f_{pay}, we can approximate the initial state $\left|\bar{\boldsymbol{V}}(0)\right\rangle$ using an appropriate ansatz. To show that the initial state can be approximated by $|\nu(\boldsymbol{\theta}_0)\rangle = \alpha_0 R_0(\boldsymbol{\theta}_0) |0\rangle$, where $\alpha_0 = \sqrt{\sum_k f_{\mathrm{pay}}(\boldsymbol{x}^{(k)})^2}$ and $R_0(\boldsymbol{\theta}_0)$ is the ansatz shown in Fig. 5.1, we adopt L-BFGS-B to find $\boldsymbol{\theta}_0$ such that

$$\max_{\boldsymbol{\theta}_0} |\left\langle \bar{\boldsymbol{V}}(0) \middle| \nu(\boldsymbol{\theta}_0) \right\rangle|^2, \tag{5.57}$$

with SciPy [112]. For the calculation of the gradient, we use the parameter shift rule [46]. We choose the parameters as $K = 1, l = 0.5, u = 2$ and the ansatz with 6 qubits and 6 layers. By doing maximization of Eq. (5.57), the value $\alpha^{-2} |\left\langle \bar{\boldsymbol{V}}(0) \middle| \nu(\boldsymbol{\theta}_0) \right\rangle|^2$, which corresponds to fidelity, should asymptotically converge to 1. The result for a payoff function of the single asset call option $f_{\mathrm{pay}}(x) = \max(x - K, 0)$ is shown in Fig. 5.4. We can see that the ansatz approximates the payoff function well. Indeed, the result satisfies $|1 - \alpha_0^{-2}| \left\langle \bar{\boldsymbol{V}}(0) \middle| \nu(\boldsymbol{\theta}_0) \right\rangle|^2| \leq 1.2 \times 10^{-5}$.

Note that this optimization does not correspond to real physical operations. What we show is that there exists $\boldsymbol{\theta}_0$ that at least approximates $\left|\bar{\boldsymbol{V}}(0)\right\rangle$ well, and we leave the efficient search algorithm for such $\boldsymbol{\theta}_0$ to future work.

5.5 Conclusion

In this chapter, we simulate the BSPDE by VQS and obtain the state which embeds the solution of the BSPDE $|\boldsymbol{V}(t_{\text{ter}})\rangle$ at t_{ter}, and utilizing the fact that the derivative price is a martingale, we calculate the derivative price by the inner product of the state $|\boldsymbol{V}(t_{\text{ter}})\rangle$ and the state $|\boldsymbol{p}(t_{\text{ter}})\rangle$ which embeds the probability distribution. Although it is difficult to accurately estimate the complexity due to the heuristic nature of variational quantum computation, at least in the numerical simulation, we confirm that the proposed method can be performed for the one-asset double barrier option and that the derivative price can be obtained with better accuracy by increasing the number of qubits and the number of layers of ansatz. We see that the computational complexity is obtained by Table 5.1 under certain assumptions, and the complexity with respect to ϵ is $O(1/\epsilon^2 \left(\log(1/\epsilon)\right)^d)$. This means that there would be a significant improvement compared to the classical FDM and conventional quantum algorithms whose complexity has factors like $(1/\epsilon)^{O(d)}$. Furthermore, we show that an oracle that generates an initial state with embedded payoff functions for typical payoff functions could be represented using an appropriate ansatz.

In this paper, we simply assumed that the initial state of the BSPDE and the state with embedded probability distribution are effectively generated by some variational quantum algorithms. We will confirm this point in future work.

Chapter 6

Conclusion

In this thesis, we have proposed a quantum algorithm that efficiently computes derivative prices via efficient classical-quantum information conversion. Although quantum amplitude estimation and quantum ordinary differential equation solvers are employed to speed up the computation of derivatives pricing, there may exist bottlenecks in the transformation from classical to quantum information and from quantum information to classical information. By alleviating these bottlenecks, our algorithm will be possible to efficiently calculate derivative prices. This would enable complex calculations of derivative prices that could not be performed before and also enable real-time calculations of the present price of the derivatives.

In Chap. 3, we have shown that a variational quantum calculation can be used to simulate the time evolution of a probability distribution to embed the classical probability distribution of the solution of a stochastic differential equation. This allows the embedding of classical information into the quantum state, which is needed for quantum algorithms such as quantum amplitude estimation. In other quantum state generation models such as quantum generative adversarial networks, it is necessary to solve an optimization problem at each time point to embed probability distributions at multiple time points. Our method has the advantage that we can embed probability distributions at multiple time points in a single simulation.

In Chap. 4, we have shown an algorithm for derivative pricing using FDM on a fault-tolerant quantum computer. In this algorithm, we proposed a method to calculate the present price of the derivative from the inner product of the quantum states, using the fact that the present price of the derivative can be approximated using the expected value of the derivative price at any future time. After the discretization of the Black-Scholes equation with FDM, the quantum ordinary differential equation solver is used to solve the equation to a certain time in the future. In addition, we prepare a quantum state in which the probability distribution at that point in time is embedded. By calculating the inner product of the quantum

state of the solution of the quantum ordinary differential equation solver and the quantum state in which the probability distribution is embedded using quantum amplitude estimation, we can approximate the present price of the derivative. Although classical FDM requires a computational complexity of $O((1/\epsilon)^{O(d)})$ to compute the derivative price of a d asset with an accuracy ϵ, this quantum algorithm requires only $\mathrm{poly}(1/\epsilon, d)$, which means that significant speedup is achieved.

In Chap. 5, we have proposed a variational quantum algorithm for derivative pricing exploiting the key idea of Chap. 4. Instead of using the quantum ordinary differential equation solver and quantum amplitude estimation, we use variational quantum simulation and the SWAP test. This enables derivative pricing by FDM even on noisy intermediate-scale quantum computers. Although it is difficult to accurately estimate the computational complexity of the algorithm due to the nature of variational computation, at least in the numerical simulation, we have confirmed that our algorithm is feasible and could reduce the complexity compared to the classical FDM.

Since derivatives price calculation is a practically important issue, the speedup achieved by quantum algorithms will have a significant impact, and further research will be accelerated in the future. The application of quantum algorithms to financial engineering has just begun, and there are still a lot of challenges. One of these is the evaluation of X-value adjustments (XVA), which is a generic term for complex risk assessment adjustments that take into account defaults, financial regulations, etc. These are problems similar to derivative pricing with more complex structures that require greater computational resources. The same bottleneck in the transformation of classical-quantum information would also exist for these problems. Furthermore, stochastic analysis, especially martingales, plays an important role in financial engineering other than derivatives pricing, and thus our algorithm may be effective in this regard. Numerical calculation of expected values appears in various fields other than finance. Since it is necessary to embed probability distributions into quantum states to compute expected values by quantum computation, we hope that our algorithm will be utilized beyond finance. Further investigation of more efficient quantum algorithms will also be a future task.

List of Activities

1 Papers

- K. Kubo, Y. O. Nakagawa, S. Endo, and S. Nagayama, “Variational quantum simulations of stochastic differential equations”, Physical Review A 103 (5), 052425 (2021)

- K. Miyamoto, K. Kubo, “Pricing multi-asset derivatives by finite difference method on a quantum computer”, IEEE Transactions on Quantum Engineering 3, 3100225 (2021)

- K. Kubo, K. Miyamoto, K. Mitarai, K. Fujii, “Pricing multi-asset derivatives by variational quantum algorithms”, arXiv:2207.01277

- N. Shirai, K. Kubo, K. Mitarai, and K. Fujii, “Quantum tangent kernel”, arXiv:2111.02951

2 Presentations and awards

- K. Kubo, Y. O. Nakagawa, S. Endo, and S. Nagayama, “Variational quantum simulations of stochastic differential equations”, Asian Conference on Quantum Information Science 2021, Online Japan (2021)

- 久保健治, 永山翔太, “デリバティブ価格決定問題への NISQ アルゴリズムの応用可能性と課題”, 第 41 回量子情報技術研究会 (2019)

- 久保健治, “金融分野への量子コンピュータの応用”, Q-LEAP 分野横断ワークショップ 量子コンピュータ研究開発の現在とこれから (2020)

- 久保健治, 中川 裕也, 遠藤傑, 永山翔太, “変分量子計算による確率微分方程式のシミュレーション” 第 1 回量子ソフトウェア研究発表会 (2020) 学生奨励賞受賞

- 久保健治, 宮本幸一, “変分量子シミュレーションによる多資産デリバティブの価格決定”, 第 45 回量子情報技術研究会 (2021)

- 宮本幸一, 久保健治, “Pricing Multi-asset Derivatives by Finite Difference Method on a Quantum Computer”, 第 56 回 2021 年度冬季 JAFEE 大会 (2022)
- 宮本幸一, 久保健治, “Pricing multi-asset derivatives by finite difference method on a quantum computer”, 第 5 回量子ソフトウェア研究発表会 (2022)

3 Patents

- 宮本幸一, 久保健治, “量子コンピュータを用いた有限差分法によるマルチアセットデリバティブの時価評価手法”, 特許出願番号 2021-154441

Appendix

A1 Variational quantum simulation of the stochastic differential equation

A1.1 Complexity of calculating expectation value

This Appendix derives the computational complexity of calculating the expectation by Eq. (3.24). To limit the error ϵ in expectation value $E = \sqrt{\left\langle \tilde{\psi} \right| S_f \left|0\right\rangle \left\langle 0\right| S_f^\dagger \left| \tilde{\psi} \right\rangle}$, we show the upper limit of error ϵ' of the expectation value for each term in Eq. (3.25), and find the number of measurements and gate complexity required to achieve this error. We assume that $S_f \left|0\right\rangle \left\langle 0\right| S_f^\dagger$ can be written as a linear combination of N_U unitary operators as follows

$$S_f \left|0\right\rangle \left\langle 0\right| S_f^\dagger = \sum_{i=1}^{N_U} \beta_i U_i, \tag{A1}$$

where $\{U_i\}$ are unitary operators. We denote the error of expectation values $\left\langle \psi\right| U_i \left|\psi\right\rangle$, where $\left|\psi\right\rangle$ is the normalized state,

$$\left|\psi\right\rangle = \frac{1}{\sqrt{\sum_{j=0}^{2^n-1} p_j^2}} \left| \tilde{\psi} \right\rangle, \tag{A2}$$

where $p_j = \mathrm{Prob}[X(t) = x_i]$. We define the error as ϵ' of the expectation value of each term in a state $\left|\psi\right\rangle$. That is, the estimated expectation value of each term $\tilde{u}_i$ satisfies

$$\left|\tilde{u}_i - \left\langle \psi\right| U_i \left|\psi\right\rangle\right| \leq \epsilon'. \tag{A3}$$

The error in the linear combination of expectation values is determined as

$$\begin{aligned}\left|\sum_{i=1}^{N_U}\beta_i\tilde{u}_i - \langle\psi|\sum_{i=1}^{N_U}\beta_i U_i|\psi\rangle\right| &= \left|\sum_{i=1}^{N_U}\beta_i\left(\tilde{u}_i - \langle\psi|U_i|\psi\rangle\right)\right| \\ &\leq \sum_{i=1}^{N_U}|\beta_i|\left|\left(\tilde{u}_i - \langle\psi|U_i|\psi\rangle\right)\right| \\ &\leq \epsilon'\sum_{i=1}^{N_U}|\beta_i|\,. \end{aligned} \tag{A4}$$

Denoting the estimation of E as $\tilde{E}$, we have

$$\begin{aligned}\left|\tilde{E}-E\right| &\leq \sum_{j=0}^{2^n-1}p_j^2\frac{\epsilon'\sum_{i=1}^{N_U}|\beta_i|}{\tilde{E}+E} \\ &\sim \sum_{j=0}^{2^n-1}p_j^2\frac{\epsilon'\sum_{i=1}^{N_U}|\beta_i|}{2E}. \end{aligned} \tag{A5}$$

To upper bound the error ϵ in E, the error ϵ' must satisfy following condition:

$$\epsilon' \lesssim \frac{2\epsilon E}{\sum_{j=0}^{2^n-1}p_j^2\sum_{i=1}^{N_U}|\beta_i|} = \gamma\epsilon, \tag{A6}$$

where $\gamma \equiv \left(2E/\sum_{j=0}^{2^n-1}p_j^2\cdot\sum_{i=1}^{N_U}|\beta_i|\right)$.

The Hadamard test (Fig. 3.2) requires $O(1/\epsilon'^2)$ measurements to limit the error in the expectation value to ϵ', and the depth of the quantum circuit is $O(1)$ in terms of the unitary $U(=Q_iQ_{i'}^\dagger, Q_iC^nZ\cdot X^{\otimes n}Q_{i'}^\dagger)$ except for the circuit to prepare the quantum state. On the other hand, the number of measurements to achieve the same accuracy with QPE is $O(\log(1/\epsilon'))$, but the depth of the quantum circuit in terms of the unitary U is $O(1/\epsilon')$ [57, 58]. The total number of measurements is equal to the number of measurements for each term multiplied by N_U. Note that $N_U = O(d^2n^{2L+2})$ with Lth-order piecewise polynomial approximation of the function f with d intervals (see Sec. 3.4.2). Therefore, the total number of measurements required to calculate the expectation value is $O(d^2n^{2L+2}/\gamma\epsilon^2)$ by Hadamard test and $O(d^2n^{2L+2}\log(1/\gamma\epsilon))$ by QPE. QPE requires extra U gates, the number of which is $O(1/\gamma\epsilon)$.

Finally, we provide the estimation of the value of γ as follows. The factor $\sum_{j=0}^{2^n-1}p_j^2$ satisfies $\sum_{j=0}^{2^n-1}p_j^2 \leq 1$ and then $\gamma \geq 2E/\sum_{i=1}^{N_U}|\beta_i|$. To evaluate $\sum_{i=1}^{N_U}|\beta_i|$, we use Eq. (3.25) and obtain

$$\sum_{i=1}^{N_U}|\beta_i| = 2\sum_{l,l'}|\xi_l\xi_{l'}^*| = 2\sum_{l,l'}|\xi_l||\xi_{l'}|. \tag{A7}$$

We estimate the upper limit of sum of absolute values of coefficients in Eq. (3.35) to evaluate $\sum_l |\xi_l|$. From Eq. (3.18), the absolute values of coefficients of $(D(n))^m$ is at most $O(2^{nm}) = O(x_{\max}^m)$. As $S_{\chi_{\alpha_k}}$ is a linear combination of $S_{\chi_l^\alpha}$, the absolute values of coefficients of $S_{\chi_{\alpha_k}}$ is at most $O(2^{(k_a-1)/2})$ from Eq. (3.33). Since k_a satisfies $0 < k_a \leq n$ by definition, $O(2^{(k_a-1)/2}) = O(\sqrt{x_{\max}})$. The largest $|\xi_l|$ is as large as $O\left(\left(\max_k \max_m \left(|a_m^{(k)}| x_{\max}^m\right)\right)\sqrt{x_{\max}}\right)$ from Eq. (3.35). Thus, we obtain

$$\sum_{i=1}^{N_U} |\beta_i| \lesssim O\left(\left[\max_k \max_m |a_m^{(k)}| x_{\max}^m\right]^2 d^2 n^{2L+2} x_{\max}\right). \tag{A8}$$

Therefore, γ is larger than the ratio of E and the right-hand side of Eq. (A8).

A1.2 Definition and construction of the tree-model approximation

Let us consider a SDE with D variables,

$$dX_d(t) = \mu_d(X_d, t)dt + \sigma_d(X_d, t)dW_d, \tag{A9}$$

for $X_1(t), \ldots, X_D(t)$, where $\{W_d\}_{d=1}^D$ describes the Brownian motion with correlation $\mathrm{Corr}[W_k, W_l] = \rho_{kl}$. For simplicity, we assume an event space of each variable $X_d(t)$ as $[0, x_{\max}^{(d)}]$, and divide it into $N_x + 1$ points; that is, $x_i^{(d)} \equiv i\Delta x^{(d)}, \Delta x^{(d)} \equiv x_{\max}^{(d)}/N_x$ $(d = 1, \ldots, D)$. The time period of the simulation, $t \in [0, T]$, is divided into $N_t + 1$ points, $t_j \equiv j\Delta t$; that is $\Delta t \equiv T/N_t$.

We define a lattice of the tree model for Eq. (A9) with nodes $(i_1, \ldots, i_D; j)$ representing the random variables $(X_1(t_j), \ldots, X_D(t_j)) = (x_{i_1}^{(1)}, \ldots, x_{i_D}^{(D)})$, where $i_d = 0, \ldots, N_x$, $j = 0, \ldots, N_t$, $d = 1, \ldots, D$. The node transitions during time $t_j \to t_{j+1}$ are of three types:

$$\begin{aligned}
&(1)\ (i_1, \ldots, i_D; j) &&\to\ (i_1, \ldots, i_D; j+1),\\
&(2)\ (i_1, \ldots, i_D; j) &&\to\ (i_1, \ldots, i_k \pm 1, \ldots, i_D; j+1),\\
&(3)\ (i_1, \ldots, i_D; j) &&\to\ (i_1, \ldots, i_k \pm 1, \ldots, i_l \pm 1, \ldots, i_D; j+1),
\end{aligned}$$

where $1 \leq k < l \leq D$. Type (1), (2) and (3) transitions occur to nodes with identical variable values, to nodes where one-variable X_k hops to its adjacent values, and to nodes where two variables (X_k and X_l) hop to their adjacent values, respectively. The transition probabilities associated with type (1), (2) and (3) transitions are respectively given by

$$\begin{aligned}
&p_m(x_{i_1}^{(1)}, \ldots, x_{i_D}^{(D)}, t),\\
&p_{u,d}^{(k)}(x_{i_1}^{(1)}, \ldots, x_{i_D}^{(D)}, t),\\
&p_{uu,ud,du,dd}^{(k,l)}(x_{i_1}^{(1)}, \ldots, x_{i_D}^{(D)}, t),
\end{aligned}$$

where the subscript $u(d)$ corresponds to the sign $+(-)$.

The transition probabilities can be determined identically to those of the one-variable SDE. The SDE (A9) at at $(X_1(t_j), \ldots, X_D(t_j)) = (x_{i_1}^{(1)}, \ldots, x_{i_D}^{(D)})$ is discretized as

$$X_d(t_{j+1}) - X_d(t_j) = \mu_d(X_d(t_j), t)\Delta t + \sigma_d(X_d(t_j), t)\sqrt{\Delta t} z_d, \tag{A10}$$

where $\{z_d\}_{d=1}^D$ is sampled from the multi-variable Gaussian distribution, $\mathrm{E}[z_d] = 0, \mathrm{Var}[z_d] = 1, \mathrm{Corr}[z_k, z_k] = \rho_{kl}$. The first and second first and second conditional moments satisfy

$$E[X_d(t_{j+1}) - X_d(t_j)|X_d(t_j) = x] \quad = \quad \mu_d(x, t_j)\Delta t, \tag{A11}$$

$$\mathrm{Var}[X_d(t_{j+1}) - X_d(t_j)|X_d(t_j) = x] \quad = \quad \sigma_d^2(x, t_j)\Delta t \tag{A12}$$

for $d = 1, \ldots, D$ and the covariance of the variables satisfies

$$\begin{aligned} &\mathrm{Cov}[X_k(t_{j+1}) - X_k(t_j), X_l(t_{j+1}) - X_l(t_j) \\ &\quad |X_k(t_j) = x, X_l(t_j) = y] \\ &=\sigma_k(x, t_j)\sigma_l(y, t_j)\rho_{kl}\Delta t \end{aligned} \tag{A13}$$

for $1 \le k < l \le D$. The corresponding quantities in the tree model are

$$\begin{aligned} &E[X_d(t_{j+1}) - X_d(t_j)|X_d(t_j) = x] \\ = \quad &\left(p_u^{(d)} - p_d^{(d)} + \sum_{k=1}^{d-1} \left(p_{uu}^{(k,d)} - p_{ud}^{(k,d)} + p_{du}^{(k,d)} - p_{dd}^{(k,d)} \right) \right. \\ &\left. + \sum_{l=d+1}^{D} \left(p_{uu}^{(d,l)} + p_{ud}^{(d,l)} - p_{du}^{(d,l)} - p_{dd}^{(d,l)} \right) \right) \Delta x^{(d)} \end{aligned} \tag{A14}$$

$$\begin{aligned} &\mathrm{Var}[X_d(t_{j+1}) - X_d(t_j)|X_d(t_j) = x] \\ = \quad &\left(p_u^{(d)} + p_d^{(d)} + \sum_{k=1}^{d-1} \left(p_{uu}^{(k,d)} + p_{ud}^{(k,d)} + p_{du}^{(k,d)} + p_{dd}^{(k,d)} \right) \right. \\ &\left. + \sum_{l=d+1}^{D} \left(p_{uu}^{(d,l)} + p_{ud}^{(d,l)} + p_{du}^{(d,l)} + p_{dd}^{(d,l)} \right) \right) \left(\Delta x^{(d)} \right)^2 \end{aligned} \tag{A15}$$

for $d = 1, \ldots, D$ and

$$\begin{aligned} &\mathrm{Cov}[X_k(t_{j+1}) - X_k(t_j), X_l(t_{j+1}) - X_l(t_j) \\ &\quad |X_k(t_j) = x, X_l(t_j) = y] \\ &= \left(p_{uu}^{(k,l)} - p_{ud}^{(k,l)} - p_{du}^{(k,l)} + p_{dd}^{(k,l)} \right) \Delta x^{(k)} \Delta x^{(l)}. \end{aligned} \tag{A16}$$

As is the same for the case of a single variable we set the transition amplitudes by equating Eqs. (A11),(A12),(A13) with (A14),(A15),(A16). If the solutions of $p^{(k)}_{u,d}, p^{(k,l)}_{uu,ud,du,dd}$ are proportional to Δt, the linear differential equation can be derived by taking the limit of $\Delta t \to 0$ (as in the one-dimensional case Eq. (3.10)).

When $D > 1$, one should note the numbers of variables and conditional expressions. As the numbers of p_m, $p^{(k)}_{u,d}, p^{(k,l)}_{uu,ud,du,dd}$ are $1, 2D, 2D(D-1)$, respectively, the number of independent variables is $2D^2$ under the normalized probability conditions. On the other hand, the number of equations of the mean, variance, and covariance are $D, D, D(D-1)/2$, respectively, so the total number of equations is $D(D+3)/2$. When $D > 1$, the number of variables exceeds the number of conditions, so an infinite number of transition probabilities satisfy the condition.

Here, we show there is indeed a solution of the transition amplitudes which admit taking limit $\Delta t \to 0$ and obtain the linear differential equitation of the probability distributions of the SDE. Fixing $p^{(k)}_{dd} = p^{(k)}_{ud} = p^{(k)}_{du} = 0$, the number of variables becomes $D(D+3)/2$, which is slightly asymmetric (because we consider only p^{k}_{uu} to be nonzero), but agrees with the number of conditional expressions. In this case, the transition probabilities are

$$p^{(k,l)}_{uu} = \frac{\sigma_k \sigma_l \rho_{kl}}{\Delta x^{(k)} \Delta x^{(l)}} \Delta t, \tag{A17}$$

$$p^{(d)}_{u} = \frac{1}{2}\left(\frac{\sigma_d^2}{\left(\Delta x^{(d)}\right)^2} + \frac{\mu_d}{\Delta x^{(d)}} \right) - \sum_{k \neq d} \frac{\sigma_k \sigma_d \rho_{kd}}{\Delta x^{(k)} \Delta x^{(d)}}, \tag{A18}$$

$$p^{(d)}_{d} = \frac{1}{2}\left(\frac{\sigma_d^2}{\left(\Delta x^{(d)}\right)^2} - \frac{\mu_d}{\Delta x^{(d)}} \right), \tag{A19}$$

$$\begin{aligned} p_m &= 1 - \left[\sum_{d=1}^{D} \left(\frac{\sigma_d^2}{\left(\Delta x^{(d)}\right)^2} - \sum_{k \neq d} \frac{\sigma_k \sigma_d \rho_{kd}}{\Delta x^{(k)} \Delta x^{(d)}} \right) \right. \\ &\quad \left. - \sum_{k \neq l} \frac{\sigma_k \sigma_l \rho_{kl}}{\Delta x^{(k)} \Delta x^{(l)}} \right] \Delta t \\ &= 1 - \sum_{d=1}^{D} \frac{\sigma_d^2}{\left(\Delta x^{(d)}\right)^2} \Delta t. \end{aligned} \tag{A20}$$

Here, we omit the arguments of μ_d and σ_d to simplify the notation.

A1.3 Mapping to VQS and construction of $L(t)$

In the multivariate case, we can construct $L(t)$ as described in Sec. 3.3. For notational simplicity, we denote $|i_1, \ldots, i_D\rangle = |\boldsymbol{i}\rangle$,$|i_1, \ldots, i_{d-1}, i_d \pm 1, i_{d+1}, \ldots, i_D\rangle =$

$|\boldsymbol{i} \pm \boldsymbol{e}_d\rangle, |i_1, \ldots, i_k + 1, \ldots, i_l + 1, \ldots i_D\rangle = |\boldsymbol{i} + \boldsymbol{e}_k + \boldsymbol{e}_l\rangle$. Using Eqs. (A17) (A18) (A19) and (A20), we obtain

$$
\begin{aligned}
L(t) &= \frac{1}{2}\sum_{d=1}^{D}\sum_{i_d=0}^{2^n-2}\sum_{i_{-d}}\left[\frac{\sigma_d^2}{\left(\Delta x^{(d)}\right)^2} + \frac{\mu_d}{\Delta x^{(d)}}\right. \\
&\quad \left. - \sum_{k\neq d}\frac{\sigma_k\sigma_d\rho_{kd}}{\Delta x^{(k)}\Delta x^{(d)}}\right] |\boldsymbol{i} + \boldsymbol{e}_d\rangle \langle \boldsymbol{i}| \\
&+ \frac{1}{2}\sum_{d=1}^{D}\sum_{i_d=1}^{2^n-1}\sum_{i_{-d}}\left(\frac{\sigma_d^2}{\left(\Delta x^{(d)}\right)^2} - \frac{\mu_d}{\Delta x^{(d)}}\right) \\
&\quad \times |\boldsymbol{i} - \boldsymbol{e}_d\rangle \langle \boldsymbol{i}| \\
&+ \sum_{k\neq l}\sum_{i_{k,l}=0}^{2^n-2}\sum_{i_{-k},i_{-l}}\frac{\sigma_k\sigma_l\rho_{kl}}{\Delta x^{(k)}\Delta x^{(l)}} \\
&\quad \times |\boldsymbol{i} + \boldsymbol{e}_k + \boldsymbol{e}_l\rangle \langle \boldsymbol{i}| \\
&- \sum_{k=1}^{D}\sum_{\boldsymbol{i}}\frac{\sigma_d^2}{\left(\Delta x^{(d)}\right)^2} |\boldsymbol{i}\rangle \langle \boldsymbol{i}| \qquad \text{(A21)}
\end{aligned}
$$

where $\sum_{\boldsymbol{i}}$ denotes the sum of $i_m \in \{0, \ldots, 2^n - 1\}$ for all $m \in \{1, \ldots, D\}$, $\sum_{i_{-d}}$ is the sum of $i_m \in \{0, \ldots, 2^n - 1\}$ for all $m \neq d$, and $\sum_{i_{-k,-l}}$ is the sum for $i_m \in \{0, \ldots, 2^n - 1\}$ for all $m \neq k, l$.

Here, we expand $\sigma_k(x^{(k)}, t), \mu_k(x^{(k)}, t)$ as

$$
\begin{aligned}
\sigma_k(x^{(k)}, t) &= \sum_{m=0}^{m_{\sigma_k}} a_{\sigma,m}^{(k)}(t)(x^{(k)})^m, \qquad \text{(A22)} \\
\mu_k(x^{(k)}, t) &= \sum_{m=0}^{m_{\mu_k}} a_{\mu,m}^{(k)}(t)(x^{(k)})^m. \qquad \text{(A23)}
\end{aligned}
$$

We also define the operators

$$
\begin{aligned}
V_+^{(k)}(n) &= I^{\otimes k-1} \otimes V_+(n) \otimes I^{\otimes D-k}, \qquad \text{(A24)} \\
V_-^{(k)}(n) &= I^{\otimes k-1} \otimes V_-(n) \otimes I^{\otimes D-k}, \qquad \text{(A25)} \\
D^{(k)}(n) &= I^{\otimes k-1} \otimes D(n) \otimes I^{\otimes D-k}. \qquad \text{(A26)}
\end{aligned}
$$

These operators satisfy the following equations:

$$V_+^{(k)}(n)(D^{(k)}(n))^m = \sum_{i_k=0}^{2^n-2} \sum_{i_{-k}} i_k^m \left|\boldsymbol{i}+\boldsymbol{e}_k\right\rangle \left\langle \boldsymbol{i}\right| \tag{A27}$$

$$V_-^{(k)}(n)(D^{(k)}(n))^m = \sum_{i_k=1}^{2^n-1} \sum_{i_{-k}} i_k^m \left|\boldsymbol{i}-\boldsymbol{e}_k\right\rangle \left\langle \boldsymbol{i}\right| \tag{A28}$$

$$\begin{aligned} & V_+^{(k)}(n)(D^{(k)}(n))^{m_k} V_+^{(l)}(n)(D^{(l)}(n))^{m_l} \\ = & \sum_{i_k=1}^{2^n-2} \sum_{i_l=1}^{2^n-2} \sum_{i_{-k,-l}} i_k^{m_k} i_l^{m_l} \left|\boldsymbol{i}+\boldsymbol{e}_k+\boldsymbol{e}_l\right\rangle \left\langle \boldsymbol{i}\right| . \end{aligned} \tag{A29}$$

Using these operators, we can rewrite $L(t)$ as

$$\begin{aligned} L(t) = & \sum_{d=1}^{D} \sum_{m_k=0}^{m_{\sigma_d}} \sum_{m_l=0}^{m_{\sigma_d}} a_{\sigma,m_k}^{(d)} a_{\sigma,m_l}^{(d)} \left(\Delta x^{(d)}\right)^{m_k+m_l-2} \\ & \times \left(\frac{V_+^{(k)}+V_-^{(k)}}{2} - I\right) (D^{(d)}(n))^{m_k+m_l} \\ & + \sum_{d=1}^{D} \sum_{m_k=0}^{m_{\mu_d}} \sum_{m_l=0}^{m_{\mu_d}} a_{\mu,m_k}^{(d)} a_{\mu,m_l}^{(d)} \left(\Delta x^{(d)}\right)^{m_k+m_l-1} \\ & \times \left(\frac{V_+^{(k)}-V_-^{(k)}}{2}\right) (D^{(d)}(n))^{m_k+m_l} \\ & + \sum_{k\neq d} \sum_{m_k=0}^{m_{\sigma_k}} \sum_{m_l=0}^{m_{\sigma_l}} a_{\sigma,m_k}^{(k)} a_{\sigma,m_l}^{(l)} \left(\Delta x^{(k)} \Delta x^{(l)}\right)^{-1} \\ & \times V_+^{(k)}(n)(D^{(k)}(n))^{m_k} V_+^{(l)}(n)(D^{(l)}(n))^{m_l} . \end{aligned} \tag{A30}$$

As $V_+^{(k)}(n)(D^{(k)}(n))^m, V_-^{(k)}(n)(D^{(k)}(n))^m$ are the sums of $O(n^m)$ unitaries composed of $O(n^2)$ few-qubit gates, Eq. (A30) is feasible decomposition of $L(t)$.

A1.4 Evaluating the expectation value

To perform computation of the expectation value, we construct a multivariate indicator operator. In the D dimensional case, the domain of the function is $\prod_{i=1}^{D}[0, x_{\max}^{(i)}]$. In each dimension, we divide $[0, x_{\max}^{(i)}]$ into d intervals $\{[a_0^{(i)}, a_1^{(i)}], , \ldots, [a_{d-1}^{(i)}, x_{\max}^{(i)}]\}$ and obtain d^D regions $I(\{k_i\}) =$

$\prod_{i=1}^{D}[a_{k_i}^{(i)}, a_{k_i+1}^{(i)}]$. The indicator operator on $I(\{k_i\})$ is represented by the tensor product of the one-dimensional indicator operator Eq. (3.23), i.e.,

$$S_{\chi_{I(\{k_i\})}} = \bigotimes_{i=1}^{D} S_{\chi_{[a_{k_i}^{(i)}, a_{k_i+1}^{(i)}]}}. \tag{A31}$$

Thus, we can construct

$$S_f = \sum_{\{k_i\}} \sum_{m=0}^{m_{k_i}} a_m^{(k_i)} (D(n))^m S_{\chi_{I(\{k_i\})}}. \tag{A32}$$

Note that $S_f |0\rangle \langle 0| S_f^\dagger$ is the sum of $O(n^{2D(m+1)})$ unitaries and each Q_k in Eq. (3.25) is composed of $O(n^4)$ gates. In general, the number of sums grows exponentially with the dimensions. However, the number of sums can be kept small if only a few of those variables are needed to compute the expectation value. Thus, when the number of sums required to construct S_f is independent of the dimension D of the random variable, our algorithm may be particularly effective.

A1.5 Error from Piecewise Polynomial Approximation

In this subsection, we evaluate the error of the expectation value $E[f(X(T))]$ from the polynomial approximation of the function f.

As in the main text, we divide $[0, x_{\max}]$ into d intervals $\{[0, a_1], \ldots, [a_{d-1}, x_{\max}]\}$. For simplicity, we assume the equally-spaced intervals, so the width of the intervals is $h = x_{\max}/d$, We ignore the errors in the probability density function $p(x)$ that come from the tree model approximation of the SDE and the incompleteness of the ansatz of VQS because we focus on the error derived from the piecewise polynomial approximation of f.

We define the Lth order residual term of the Taylor expansion of f around $a_k = kh$ as

$$R_k^L(x) = \frac{1}{(L+1)!} f^{(n)}(c)(x - kh)^{L+1}, \tag{A33}$$

where $x \in [a_k, a_{k+1}] = [kh, (k+1)h]$ and $c \in [x, (k+1)h]$. As $x - kh \leq h$, $R_k^L(x)$ is $O(h^{L+1})$. When we approximate f on $[a_k, a_{k+1}]$ by the Lth order Taylor expansion $g^L(x)$, the error of expectation value $E_f =$

$\sum_{i=0}^{2^n-1} f(x_i)p(x_i)$ is

$$\begin{aligned}
&|E_f - E_g| \\
&= \left| \sum_{k=1}^{d-1} \int_{kh}^{(k+1)h} f(x)p(x)dx - \sum_{k=1}^{d-1} \int_{kh}^{(k+1)h} g^L(x)p(x)dx \right| \\
&= \left| \sum_{k=1}^{d-1} \int_{kh}^{(k+1)h} R_k^L(x)p(x)dx \right| \\
&\leq \max_k \left[\max_{kh \leq x \leq (k+1)h} \left(\left| R_k^L(x) \right| \right) \right] \cdot \sum_{k'=1}^{d-1} \int_{k'h}^{(k'+1)h} p(x)dx \\
&= \max_k \left[\max_{kh \leq x \leq (k+1)h} \left(\left| R_k^L(x) \right| \right) \right] \\
&= O(h^{L+1})
\end{aligned}$$

To suppress the error below ϵ, it is necessary to set $d > x_{\max} \epsilon^{-\frac{1}{L+1}}$. From the discussion in Sec. IV, $S_f |0\rangle \langle 0| S_f^\dagger$ is the sum of $O(d^2 n^{2L+2})$ unitaries. Thus, we can see that $S_f |0\rangle \langle 0| S_f^\dagger$ is the sum of $O(x_{\max}^2 \epsilon^{-\frac{2}{L+1}} n^{2L+2})$ unitaries.

A2 Pricing multi-asset derivatives by finite difference method on a quantum computer

A2.1 Proof of Lemma 2.2.1

First, we prove the following property of F in (2.25).

Lemma A2.1. *For F in (2.25), the logarithmic norm satisfies $\mu(F) < 0$.*

Proof. Since the matrix $(\rho_{ij})_{1 \leq i,j \leq d}$ is positive-definite, so is the matrix $(\sigma_i \sigma_j \rho_{ij})_{1 \leq i,j \leq d}$. Then, as mentioned in the proof of Theorem 5.1 in [27], $\mu(F^{\text{2nd}}) < 0$. Besides, since F^{1st} is anti-symmetric, $\mu(F^{\text{1st}}) = 0$. Combining this,

$$\mu(F) \leq \mu(F^{\text{1st}}) + \mu(F^{\text{2nd}}) < 0. \tag{A34}$$

□

Using this, we can prove Lemma 2.2.1.

Proof of Lemma 2.2.1. Because of (2.28), for $i \in [d]$,

$$\left| \frac{\partial}{\partial x_i} Y(\tau, \boldsymbol{x}) - \frac{Y(\tau, \boldsymbol{x} + h_i \boldsymbol{e}_i) - Y(\tau, \boldsymbol{x} - h_i \boldsymbol{e}_i)}{2h_i} \right| < \frac{\zeta}{6} h_i^2, \tag{A35}$$

and

$$\left|\frac{\partial^2}{\partial x_i^2}Y(\tau,\boldsymbol{x})-\frac{Y(\tau,\boldsymbol{x}+h_i\boldsymbol{e}_i)-2Y(\tau,\boldsymbol{x})+Y(\tau,\boldsymbol{x}-h_i\boldsymbol{e}_i)}{h_i^2}\right|<\frac{\xi}{12}h_i^2. \tag{A36}$$

hold, and, for $i,j\in[d]$ such that $i\neq j$,

$$\begin{aligned}&\left|\frac{\partial^2}{\partial x_i\partial x_j}Y(\tau,\boldsymbol{x})-\frac{1}{4h_ih_j}\left(Y(\tau,\boldsymbol{x}+h_i\boldsymbol{e}_i+h_j\boldsymbol{e}_j)-Y(\tau,\boldsymbol{x}+h_i\boldsymbol{e}_i-h_j\boldsymbol{e}_j)\right.\right.\\&\left.\left.-Y(\tau,\boldsymbol{x}-h_i\boldsymbol{e}_i+h_j\boldsymbol{e}_j)+Y(\tau,\boldsymbol{x}-h_i\boldsymbol{e}_i-h_j\boldsymbol{e}_j)\right)\right|<\frac{\xi}{6}h_ih_j\end{aligned} \tag{A37}$$

holds, where $\boldsymbol{e}_i, i\in[d]$ is the d-dimensional vector whose i-th element is 1 and the others are 0. Therefore, we see that

$$\begin{aligned}|\mathcal{L}Y(\tau,\boldsymbol{x}^{(k)})-(F\boldsymbol{Y}(\tau)+\boldsymbol{C}(\tau))_k| &< \sum_{i=1}^{d}\frac{\xi}{24}\sigma_i^2h_i^2+\sum_{i=1}^{d}\sum_{j=i+1}^{d}\frac{\xi}{6}\sigma_i\sigma_j|\rho_{ij}|h_ih_j\\&+\sum_{i=1}^{d}\frac{\zeta}{6}\left|r-\frac{1}{2}\sigma_i^2\right|h_i^2\\&<\frac{\epsilon}{T},\end{aligned} \tag{A38}$$

where $(F\boldsymbol{Y}(\tau)+\boldsymbol{C}(\tau))_k$ is the k-th element of $F\boldsymbol{Y}(\tau)+\boldsymbol{C}(\tau)$ and we used (2.29). Since

$$\frac{d}{d\tau}(\tilde{\boldsymbol{Y}}(\tau)-\boldsymbol{Y}(\tau))=F\tilde{\boldsymbol{Y}}(\tau)+\boldsymbol{C}(\tau)-\mathcal{L}\boldsymbol{Y}(\tau)=F(\tilde{\boldsymbol{Y}}(\tau)-\boldsymbol{Y}(\tau))+F\boldsymbol{Y}(\tau)+\boldsymbol{C}(\tau)-\mathcal{L}\boldsymbol{Y}(\tau), \tag{A39}$$

where $\mathcal{L}\boldsymbol{Y}(\tau):=(\mathcal{L}Y(\tau,\boldsymbol{x}^{(1)}),...,\mathcal{L}Y(\tau,\boldsymbol{x}^{(N_{\mathrm{gr}})}))^T$, we finally obtain

$$\begin{aligned}\|\tilde{\boldsymbol{Y}}(\tau)-\boldsymbol{Y}(\tau)\|&\leq e^{\tau\mu(F)}\|\tilde{\boldsymbol{Y}}(0)-\boldsymbol{Y}(0)\|\\&+\int_0^\tau e^{(\tau-\tau')\mu(F)}\|F\boldsymbol{Y}(\tau')+\boldsymbol{C}(\tau')-\mathcal{L}\boldsymbol{Y}(\tau')\|d\tau'\\&\leq\sqrt{N_{\mathrm{gr}}}\epsilon\end{aligned} \tag{A40}$$

by (2.1) in [113]. Here, we used $\tilde{\boldsymbol{Y}}(0)=\boldsymbol{Y}(0), \mu(F)<0$ and (A38).

□

A2.2 Proof of Lemma 4.3.1

Upper bound the probability that the underlying asset prices reach the boundaries

In order to prove Lemma 4.3.1, we weed some subsidiary lemmas. First, we prove the following one on the probability that the underlying asset prices reach the boundaries.

Lemma A2.2. *Let ϵ be a positive real number. For $S_i, i \in [d]$ in (2.2) and any $t \in (0, \tilde{t}_{\mathrm{u}}]$, where*

$$\tilde{t}_{\mathrm{u}} := \frac{\left(\log\left(\frac{H_i}{S_{i,0}}\right)\right)^2}{\sigma^2 \log \epsilon^{-1}}, \tag{A41}$$

the following holds

$$P\left(\max_{0\leq s\leq t} S_i(s) \geq H_i \;\middle|\; S_i(t) = s\right) \leq \epsilon \text{ if } s < \sqrt{S_{i,0}H_i}. \tag{A42}$$

Similarly, for any $t \in (0, \tilde{t}_{\mathrm{l}})$, where

$$\tilde{t}_{\mathrm{l}} := \frac{\left(\log\left(\frac{S_{i,0}}{L_i}\right)\right)^2}{\sigma^2 \log \epsilon^{-1}}, \tag{A43}$$

the following holds

$$P\left(\min_{0\leq s\leq t} S_i(s) \leq L_i \;\middle|\; S_i(t) = s\right) \leq \epsilon \text{ if } s > \sqrt{S_{i,0}L_i}. \tag{A44}$$

Proof. It is well-known (see e.g. [2]) that $S_i(t)$ can be written as

$$S_i(t) = S_{i,0} \exp\left(\sigma_i W_i(t) - \left(\frac{1}{2}\sigma_i^2 - r\right) t\right). \tag{A45}$$

Therefore, we see that

$$S_i(t) = H_i \Leftrightarrow B_i(t) := W_i(t) - \left(\frac{\sigma_i}{2} - \frac{r}{\sigma_i}\right) t = \frac{\log\left(\frac{H_i}{S_{i,0}}\right)}{\sigma_i} \tag{A46}$$

and

$$S_i(t) = \sqrt{S_{i,0}H_i} \Leftrightarrow B_i(t) = \frac{\log\left(\frac{H_i}{S_{i,0}}\right)}{2\sigma_i}. \tag{A47}$$

Using a formula on the distribution of the maximum of a Brownian bridge with drift (THEOREM 3.1 in [114]), we obtain

$$P\left(\max_{0\leq s\leq t} B_i(s) \geq \frac{\log\left(\frac{H_i}{S_{i,0}}\right)}{\sigma_i} \middle| B_i(t) = \frac{\log\left(\frac{H_i}{S_{i,0}}\right)}{2\sigma_i}\right) = \exp\left(-\frac{2}{t}\frac{\log\left(\frac{H_i}{S_{i,0}}\right)}{\sigma_i}\frac{\log\left(\frac{H_i}{S_{i,0}}\right)}{2\sigma_i}\right). \tag{A48}$$

Then, for $t \in (0, \tilde{t}_{\mathrm{u}}]$, we obtain (A42). The later part of the statement is proven similarly. □

Using Lemma A2.2, we can prove the following.

Lemma A2.3. *Consider $S_1, ..., S_d$ in (2.2). Let ϵ be a positive real number. Then, for any $\boldsymbol{S} := (s_1, ..., s_d)^T \in D_{\text{half}}$, where*

$$D_{\text{half}} := (\sqrt{L_1 S_{1,0}}, \sqrt{U_1 S_{1,0}}) \times \cdots \times (\sqrt{L_d S_{d,0}}, \sqrt{U_d S_{d,0}}), \tag{A49}$$

and any $t \in (0, t_b]$, where

$$t_b := \min \left\{ \frac{\left(\log\left(\frac{U_1}{S_{1,0}}\right)\right)^2}{\sigma_1^2 \log\left(\frac{2d}{\epsilon}\right)}, ..., \frac{\left(\log\left(\frac{U_d}{S_{d,0}}\right)\right)^2}{\sigma_d^2 \log\left(\frac{2d}{\epsilon}\right)}, \frac{\left(\log\left(\frac{S_{1,0}}{L_1}\right)\right)^2}{\sigma_1^2 \log\left(\frac{2d}{\epsilon}\right)}, ..., \frac{\left(\log\left(\frac{S_{d,0}}{L_d}\right)\right)^2}{\sigma_d^2 \log\left(\frac{2d}{\epsilon}\right)} \right\}, \tag{A50}$$

the following holds

$$p_{\text{NB}}(t, \boldsymbol{S}) \geq 1 - \epsilon, \tag{A51}$$

where $p_{\text{NB}}(t, \boldsymbol{S})$ is defined below (4.7).

Proof.

$$\begin{aligned}
& p_{\text{NB}}(t, \boldsymbol{S}) \\
\geq\ & P\left(\boldsymbol{S}(u) \text{ does not reach any boundaries by } t \mid \boldsymbol{S}_t = \boldsymbol{S}\right) \\
=\ & 1 - P\left(\boldsymbol{S}(u) \text{ reaches either of boundaries by } t \mid \boldsymbol{S}_t = \boldsymbol{S}\right) \\
\geq\ & 1 - \sum_{i=1}^{d} P\left(\max_{0 \leq u \leq t} S_i(u) \geq H_i \;\middle|\; S_i(t) = s_i\right) - \sum_{i=1}^{d} P\left(\min_{0 \leq u \leq t} S_i(u) \leq L_i \;\middle|\; S_i(t) = s_i\right) \\
\geq\ & 1 - d \times \frac{\epsilon}{2d} - d \times \frac{\epsilon}{2d} = 1 - \epsilon,
\end{aligned} \tag{A52}$$

where we used Lemma A2.2 at the last inequality. □

Upper bound the integral on the outside of the boundaries

Besides, we need the following lemmas, in order to upper bound the contribution from the outside of the boundaries to the integral (4.7).

Lemma A2.4. *Consider $S_i, i \in [d]$ in (2.2). Let H be a real number such that $H > S_{i,0}$ and ϵ be a positive real number satisfying*

$$\log\left(\frac{1}{2\epsilon}\right) > \frac{4}{5}\left(1 + \frac{2r}{\sigma_i^2}\right) \log\left(\frac{H}{S_{i,0}}\right). \tag{A53}$$

Then, for any $t \in (0, t_{\text{cu}})$,

$$\int_H^\infty s\phi_i(t, s)ds < \epsilon S_{i,0} e^{rt}, \int_H^\infty \phi_i(t, s)ds < \epsilon \tag{A54}$$

holds, where $\phi_i(t, s)$ is the probability density of $S_i(t)$ and

$$t_{\text{cu}} := \frac{8\left(\log\left(\frac{H}{S_{i,0}}\right)\right)^2}{25\sigma_i^2 \log\left(\frac{1}{2\epsilon}\right)}. \tag{A55}$$

Proof. Because of (A45) and the basic property of the Brownian motion, the probability density of $x_i(t) = \log S_i(t)$ is

$$\frac{1}{\sqrt{2\pi t}\sigma_i} \exp\left(-\frac{1}{2\sigma_i^2 t}\left(x - \left(r - \frac{1}{2}\sigma_i^2\right)t\right)^2\right). \tag{A56}$$

Therefore, we see that

$$\begin{aligned}
&\int_H^\infty s\phi_i(t,s)ds \\
=& \int_{\log(H/S_{i,0})}^\infty e^x \frac{1}{\sqrt{2\pi t}\sigma_i} \exp\left(-\frac{1}{2\sigma_i^2 t}\left(x - \left(r - \frac{1}{2}\sigma_i^2\right)t\right)^2\right) dx \\
=& \frac{S_{i,0}}{\sqrt{2\pi t}\sigma_i} e^{rt} \int_{\log(H/S_{i,0})}^\infty \exp\left(-\frac{1}{2\sigma_i^2 t}\left(x - \left(r + \frac{1}{2}\sigma_i^2\right)t\right)^2\right) dx \\
<& \frac{S_{i,0}e^{rt}}{2} \exp\left(-\frac{1}{2\sigma_i^2 t}\left(\log\left(\frac{H}{S_{i,0}}\right) - \left(r + \frac{\sigma_i^2}{2}\right)t\right)^2\right).
\end{aligned} \tag{A57}$$

Here, we used

$$\frac{2}{\sqrt{\pi}} \int_c^\infty e^{-y^2} dy < e^{-c^2}, \tag{A58}$$

which hold for any $c \in \mathbb{R}_+$. Besides, because of (A53) and (A55),

$$\left(r + \frac{\sigma^2}{2}\right)t < \left(r + \frac{\sigma^2}{2}\right) \frac{8\left(\log\left(\frac{H}{S_{i,0}}\right)\right)^2}{25\sigma^2 \log\left(\frac{1}{2\epsilon}\right)} < \frac{1}{5}\log\left(\frac{H}{S_{i,0}}\right) \tag{A59}$$

holds for $t \in (0, t_{\mathrm{cu}})$. Combining (A55), (A57) and (A59), we obtain

$$\int_H^\infty s\phi_i(t_{\mathrm{cu}},s)ds < \frac{S_{i,0}e^{rt}}{2} \exp\left(-\frac{1}{2\sigma_i^2 t}\frac{16}{25}\left(\log\left(\frac{H}{S_{i,0}}\right)\right)^2\right) < \epsilon S_{i,0}e^{rt} \tag{A60}$$

for $t \in (0, t_{\mathrm{cu}})$.

On the other hand,

$$\begin{aligned}
&\int_H^\infty \phi_i(t_{\mathrm{cu}},s)ds \\
=& \int_{\log(H/S_0)}^\infty \frac{1}{\sqrt{2\pi t}\sigma} \exp\left(-\frac{1}{2\sigma^2 t}\left(x - \left(r - \frac{1}{2}\sigma^2\right)t\right)^2\right) dx \\
<& \frac{1}{2} \exp\left(-\frac{1}{2\sigma^2 t}\left(\log\left(\frac{H}{S_0}\right) - \left(r - \frac{\sigma^2}{2}\right)t\right)^2\right),
\end{aligned} \tag{A61}$$

where we used (A58) again. Combining this and $\left(r-\frac{\sigma_i^2}{2}\right)t<\frac{1}{5}\log\left(\frac{H}{S_{i,0}}\right)$, which holds for $t\in(0,t_{\mathrm{cu}})$ because of (A59), we obtain

$$\int_H^\infty \phi(t_{\mathrm{cu}},s)ds<\frac{1}{2}\exp\left(-\frac{1}{2\sigma_i^2 t}\frac{16}{25}\left(\log\left(\frac{H}{S_{i,0}}\right)\right)^2\right)<\epsilon. \tag{A62}$$

□

Lemma A2.5. *Consider $S_i, i\in[d]$ in (2.2). Let L be a real number such that $L<S_{i,0}$ and ϵ be a positive real number satisfying*

$$\log\left(\frac{1}{2\epsilon}\right)>\frac{4}{5}\left(1-\frac{2r}{\sigma_i^2}\right)\log\left(\frac{S_{i,0}}{L}\right). \tag{A63}$$

Then, for any $t\in(0,t_{\mathrm{cl}})$,

$$\int_0^L s\phi_i(t,s)ds<\epsilon S_{i,0}e^{rt},\ \int_0^L \phi_i(t,s)ds<\epsilon \tag{A64}$$

holds, where $\phi_i(t,s)$ is the probability density of $S_i(t)$ and

$$t_{\mathrm{cl}}:=\frac{8\left(\log\left(\frac{S_{i,0}}{L}\right)\right)^2}{25\sigma_i^2\log\left(\frac{1}{2\epsilon}\right)}. \tag{A65}$$

Proof. Similarly to (A57), for $t\in(0,t_{\mathrm{cl}})$,

$$\begin{aligned}
&\int_0^L s\phi_i(t,s)ds\\
=&\int_{-\infty}^{\log(L/S_{i,0})} e^x\frac{1}{\sqrt{2\pi t}\sigma_i}\exp\left(-\frac{1}{2\sigma_i^2 t}\left(x-\left(r-\frac{1}{2}\sigma_i^2\right)t\right)^2\right)dx\\
=&\frac{S_{i,0}}{\sqrt{2\pi t}\sigma_i}e^{rt}\int_{-\infty}^{\log(L/S_{i,0})}\exp\left(-\frac{1}{2\sigma_i^2 t}\left(x-\left(r+\frac{1}{2}\sigma_i^2\right)t\right)^2\right)dx\\
<&\frac{S_{i,0}e^{rt}}{2}\exp\left(-\frac{1}{2\sigma_i^2 t}\left(\log\left(\frac{S_{i,0}}{L}\right)+\left(r+\frac{\sigma_i^2}{2}\right)t\right)^2\right)\\
<&\frac{S_{i,0}e^{rt}}{2}\exp\left(-\frac{1}{2\sigma_i^2 t}\left(\log\left(\frac{S_{i,0}}{L}\right)\right)^2\right)\\
<&\epsilon S_{i,0}e^{rt},
\end{aligned} \tag{A66}$$

where we used (A58) at the first inequality and (A65) at the last inequality.

On the other hand,

$$\begin{aligned}
&\int_0^L \phi(t,s)ds \\
=\ &\int_{-\infty}^{\log(L/S_{i,0})} \frac{1}{\sqrt{2\pi t}\sigma_i} \exp\left(-\frac{1}{2\sigma_i^2 t}\left(x-\left(r-\frac{1}{2}\sigma_i^2\right)t\right)^2\right)dx \\
<\ &\frac{1}{2}\exp\left(-\frac{1}{2\sigma_i^2 t}\left(\log\left(\frac{S_{i,0}}{L}\right)+\left(r-\frac{\sigma_i^2}{2}\right)t\right)^2\right), \qquad \text{(A67)}
\end{aligned}$$

where we used (A58) again. Then, for $t \in (0, t_{\rm cl})$, (A67) and

$$\left(\frac{\sigma_i^2}{2}-r\right)t < \left(\frac{\sigma_i^2}{2}-r\right)\frac{8\left(\log\left(\frac{S_{i,0}}{L}\right)\right)^2}{25\sigma_i^2\log\left(\frac{1}{2\epsilon}\right)} < \frac{1}{5}\log\left(\frac{S_{i,0}}{L}\right), \qquad \text{(A68)}$$

which follows (A63), lead to

$$\int_0^L \phi_i(t,s)ds < \frac{1}{2}\exp\left(-\frac{1}{2\sigma_i^2 t}\frac{16}{25}\left(\log\left(\frac{S_{i,0}}{L}\right)\right)^2\right) < \epsilon. \qquad \text{(A69)}$$

□

Combining these lemma, we obtain the following.

Lemma A2.6. *Consider $S_1, ..., S_d$ in (2.2) under Assumption 4.3.1. For any $\epsilon \in \mathbb{R}_+$ satisfying*

$$\log\left(\frac{\tilde{A}d(d+1)}{\epsilon}\right) > \max\left\{\frac{2}{5}\left(1-\frac{2r}{\sigma_i^2}\right)\log\left(\frac{U_i}{S_{i,0}}\right), \frac{2}{5}\left(1-\frac{2r}{\sigma_i^2}\right)\log\left(\frac{S_{i,0}}{L_i}\right)\right\}, i = 1, ..., d, \qquad \text{(A70)}$$

where $\tilde{A} = \max\{A_1\sqrt{U_1 S_{1,0}}, ..., A_d\sqrt{U_d S_{d,0}}, A_0\}$, the following holds

$$e^{-rt_{\rm ter}}\int_{\mathbb{R}_+^d \setminus D_{\rm half}} d\boldsymbol{S}\phi(t_{\rm ter}, \boldsymbol{S})V(t_{\rm ter}, \boldsymbol{S}) \le \epsilon, \qquad \text{(A71)}$$

where $t_{\rm ter}$ is defined as (4.14) and $D_{\rm half}$ is defined as (A49).

Proof. First, note that, under Assumption 4.3.1, for $\boldsymbol{S} = (S_1, ..., S_d)^T \in \mathbb{R}_+^d$,

$$\begin{aligned}
V(t_{\rm ter}, \boldsymbol{S}) &= E[e^{-r(T-t_{\rm ter})} f_{\rm pay}(\boldsymbol{S}(T))1_{\rm NB}|\boldsymbol{S}(t_{\rm ter}) = \boldsymbol{S}] \\
&\le E\left[e^{-r(T-t_{\rm ter})}\left(\sum_{i=1}^d A_i S_i(T) + A_0\right)\middle|\boldsymbol{S}(t_{\rm ter}) = \boldsymbol{S}\right] \\
&= \sum_{i=1}^d A_i S_i + A_0 e^{-r(T-t_{\rm ter})}. \qquad \text{(A72)}
\end{aligned}$$

Therefore, we obtain

$$e^{-rt_{\rm ter}}\int_{\mathbb{R}^d_+\setminus D_{\rm half}} d\boldsymbol{S}\phi(t_{\rm ter},\boldsymbol{S})V(t_{\rm ter},\boldsymbol{S}) \le \sum_{i=1}^d A_i e^{-rt_{\rm ter}}\int_{\mathbb{R}^d_+\setminus D_{\rm half}} d\boldsymbol{S}S_i\phi(t_{\rm ter},\boldsymbol{S}) + A_0 e^{-rT}\int_{\mathbb{R}^d_+\setminus D_{\rm half}} d\boldsymbol{S}\phi(t_{\rm ter},\boldsymbol{S}). \tag{A73}$$

We can evaluate $A_1 e^{-rt_{\rm ter}}\int_{\mathbb{R}^d_+\setminus D_{\rm half}} d\boldsymbol{S}S_1\phi(t_{\rm ter},\boldsymbol{S})$ as follows

$$\begin{aligned} A_1 e^{-rt_{\rm ter}}\int_{\mathbb{R}^d_+\setminus D_{\rm half}} d\boldsymbol{S}S_1\phi(t_{\rm ter},\boldsymbol{S}) &\le A_1 e^{-rt_{\rm ter}}\int_{S_1\ge\sqrt{U_1S_{1,0}}} d\boldsymbol{S}S_1\phi(t_{\rm ter},\boldsymbol{S}) \\ &+ A_1 e^{-rt_{\rm ter}}\int_{S_1\le\sqrt{L_1S_{1,0}}} d\boldsymbol{S}S_1\phi(t_{\rm ter},\boldsymbol{S}) \\ &+ \sum_{i=2}^d A_1 e^{-rt_{\rm ter}}\int_{\substack{\sqrt{L_1S_{1,0}}\le S_1\le\sqrt{U_1S_{1,0}}\\ S_i\ge\sqrt{U_iS_{i,0}}}} d\boldsymbol{S}S_1\phi(t_{\rm ter},\boldsymbol{S}) \\ &+ \sum_{i=2}^d A_1 e^{-rt_{\rm ter}}\int_{\substack{\sqrt{L_1S_{1,0}}\le S_1\le\sqrt{U_1S_{1,0}}\\ S_i\le\sqrt{L_iS_{i,0}}}} d\boldsymbol{S}S_1\phi(t_{\rm ter},\boldsymbol{S}) \\ &. \end{aligned} \tag{A74}$$

In the right hand side, the first term is $A_1 e^{-rt_{\rm ter}}\int_{\sqrt{U_1S_{1,0}}}^\infty dS_1S_1\phi_1(t_{\rm ter},S_1)$, where $\phi_i(t,S_i)$ is the marginal density of $S_i(t)$, and therefore

$$A_1 e^{-rt_{\rm ter}}\int_{S_1\ge\sqrt{U_1S_{1,0}}} d\boldsymbol{S}S_1\phi(t_{\rm ter},\boldsymbol{S}) \le \frac{\epsilon S_{1,0}A_1}{2d(d+1)\tilde{A}} \le \frac{\epsilon}{2d(d+1)} \tag{A75}$$

holds from Lemma A2.4[1]. Similarly, from Lemma A2.5, the second term is bounded as

$$A_1 e^{-rt_{\rm ter}}\int_{S_1\le\sqrt{L_1S_{1,0}}} d\boldsymbol{S}S_1\phi(t_{\rm ter},\boldsymbol{S}) \le \frac{\epsilon}{2d(d+1)}. \tag{A76}$$

On the other hand, from Lemma A2.4, we see that the third term is bounded

[1]Note that

$$\frac{8\left(\log\left(\frac{\sqrt{U_iS_{i,0}}}{S_{i,0}}\right)\right)^2}{25\sigma_i^2\log\left(\frac{2\tilde{A}d(d+1)}{\epsilon}\right)} = \frac{2\left(\log\left(\frac{U_i}{S_{i,0}}\right)\right)^2}{25\sigma_i^2\log\left(\frac{2\tilde{A}d(d+1)}{\epsilon}\right)}, \frac{8\left(\log\left(\frac{S_{i,0}}{\sqrt{L_iS_{i,0}}}\right)\right)^2}{25\sigma_i^2\log\left(\frac{2\tilde{A}d(d+1)}{\epsilon}\right)} = \frac{2\left(\log\left(\frac{S_{i,0}}{L_i}\right)\right)^2}{25\sigma_i^2\log\left(\frac{2\tilde{A}d(d+1)}{\epsilon}\right)}.$$

as

$$\begin{aligned}
& \sum_{i=2}^{d} A_1 e^{-rt_{\mathrm{ter}}} \int_{\substack{\sqrt{L_1 S_{1,0}} \le S_1 \le \sqrt{U_1 S_{1,0}} \\ S_i \ge \sqrt{U_i S_{i,0}}}} d\boldsymbol{S} S_1 \phi(t_{\mathrm{ter}}, \boldsymbol{S}) \\
\le & \sum_{i=2}^{d} A_1 \sqrt{U_1 S_{1,0}} \int_{\substack{\sqrt{L_1 S_{1,0}} \le S_1 \le \sqrt{U_1 S_{1,0}} \\ S_i \ge \sqrt{U_i S_{i,0}}}} d\boldsymbol{S} \phi(t_{\mathrm{ter}}, \boldsymbol{S}) \\
\le & \sum_{i=2}^{d} A_1 \sqrt{U_1 S_{1,0}} \int_{\sqrt{U_i S_{i,0}}}^{\infty} ds \phi_i(t_{\mathrm{ter}}, s) \\
\le & \sum_{i=2}^{d} \frac{\epsilon A_1 \sqrt{U_1 S_{1,0}}}{2d(d+1)\tilde{A}} \\
\le & \sum_{i=2}^{d} \frac{\epsilon}{2d(d+1)} \\
= & \frac{\epsilon(d-1)}{2d(d+1)}
\end{aligned} \tag{A77}$$

and, similarly, the fourth term is bounded as

$$\sum_{i=2}^{d} A_1 e^{-rt_{\mathrm{ter}}} \int_{\substack{\sqrt{L_1 S_{1,0}} \le S_1 \le \sqrt{U_1 S_{1,0}} \\ S_i \le \sqrt{L_i S_{i,0}}}} d\boldsymbol{S} S_1 \phi_{\boldsymbol{S}}(t_{\mathrm{ter}}, \boldsymbol{S}) \le \frac{\epsilon(d-1)}{2d(d+1)} \tag{A78}$$

by Lemma A2.5. In summary,

$$A_1 e^{-rt_{\mathrm{ter}}} \int_{\mathbb{R}^d_+ \setminus D_{\mathrm{half}}} d\boldsymbol{S} S_1 \phi(t_{\mathrm{ter}}, \boldsymbol{S}) \le \frac{\epsilon}{d+1} \tag{A79}$$

holds. $A_2 e^{-rt_{\mathrm{ter}}} \int_{\mathbb{R}^d_+ \setminus D_{\mathrm{half}}} d\boldsymbol{S} S_2 \phi(t_{\mathrm{ter}}, \boldsymbol{S}), ..., A_d e^{-rt_{\mathrm{ter}}} \int_{\mathbb{R}^d_+ \setminus D_{\mathrm{half}}} d\boldsymbol{S} S_d \phi(t_{\mathrm{ter}}, \boldsymbol{S})$ are bounded similarly.

On the other hand, by Lemmas A2.4 and A2.5,

$$\begin{aligned}
& A_0 e^{-rt_{\mathrm{ter}}} \int_{\mathbb{R}^d_+ \setminus D_{\mathrm{half}}} d\boldsymbol{S} \phi(t_{\mathrm{ter}}, \boldsymbol{S}) \\
\le & \sum_{i=1}^{d} \left(A_0 e^{-rt_{\mathrm{ter}}} \int_{S_i \ge \sqrt{U_i S_{i,0}}} d\boldsymbol{S} \phi(t_{\mathrm{ter}}, \boldsymbol{S}) + A_0 e^{-rt_{\mathrm{ter}}} \int_{S_i \le \sqrt{L_i S_{i,0}}} d\boldsymbol{S} \phi(t_{\mathrm{ter}}, \boldsymbol{S}) \right) \\
\le & \sum_{i=1}^{d} \frac{\epsilon A_0 e^{-rt_{\mathrm{ter}}}}{d(d+1)\tilde{A}} \\
\le & \frac{\epsilon}{d+1}
\end{aligned} \tag{A80}$$

holds.

Summing up all terms, we obtain (A71). □

Proof of Lemma 4.3.1

Then, we finally prove Lemma 4.3.1.

The proof of Lemma 4.3.1. Note that t_{ter} satisfies

$$t_{\mathrm{ter}} < \min\left\{ \frac{\left(\log\left(\frac{U_1}{S_{1,0}}\right)\right)^2}{\sigma_1^2 \log\left(\frac{2d(d+1)\tilde{\tilde{A}}}{\epsilon}\right)}, ..., \frac{\left(\log\left(\frac{U_d}{S_{d,0}}\right)\right)^2}{\sigma_d^2 \log\left(\frac{2d(d+1)\tilde{\tilde{A}}}{\epsilon}\right)}, \frac{\left(\log\left(\frac{S_{1,0}}{L_1}\right)\right)^2}{\sigma_1^2 \log\left(\frac{2d(d+1)\tilde{\tilde{A}}}{\epsilon}\right)}, ..., \frac{\left(\log\left(\frac{S_{d,0}}{L_d}\right)\right)^2}{\sigma_d^2 \log\left(\frac{2d(d+1)\tilde{\tilde{A}}}{\epsilon}\right)} \right\}, \tag{A81}$$

where $\tilde{\tilde{A}} := \max\{A_0, A_1 S_{1,0}, ..., A_d S_{d,0}\}$. Besides, we can see that

$$\begin{aligned}
& \left| V(0, \boldsymbol{S}_0) - e^{-rT} \int_{\tilde{\tilde{D}}} d\boldsymbol{x} \tilde{\phi}(t_{\mathrm{ter}}, \boldsymbol{x}) Y(\tau_{\mathrm{ter}}, \boldsymbol{x}) \right| \\
= & \left| V(0, \boldsymbol{S}_0) - e^{-rt_{\mathrm{ter}}} \int_{\hat{D}} d\boldsymbol{S} \phi(t_{\mathrm{ter}}, \boldsymbol{S}) V(t_{\mathrm{ter}}, \boldsymbol{S}) \right| \\
= & \left| e^{-rt_{\mathrm{ter}}} \int d\boldsymbol{S}_{\mathbb{R}_+^d} \phi(t_{\mathrm{ter}}, \boldsymbol{S}) p_{\mathrm{NB}}(t_{\mathrm{ter}}, \boldsymbol{S}) V(t_{\mathrm{ter}}, \boldsymbol{S}) - e^{-rt_{\mathrm{ter}}} \int_{\hat{D}} d\boldsymbol{S} \phi(t_{\mathrm{ter}}, \boldsymbol{S}) V(t_{\mathrm{ter}}, \boldsymbol{S}) \right| \\
= & \left| e^{-rt_{\mathrm{ter}}} \int_{D_{\mathrm{half}}} d\boldsymbol{S} \phi(t_{\mathrm{ter}}, \boldsymbol{S}) p_{\mathrm{NB}}(t_{\mathrm{ter}}, \boldsymbol{S}) V(t_{\mathrm{ter}}, \boldsymbol{S}) \right. \\
& \quad + e^{-rt_{\mathrm{ter}}} \int_{\mathbb{R}_+^d \setminus D_{\mathrm{half}}} d\boldsymbol{S} \phi(t_{\mathrm{ter}}, \boldsymbol{S}) p_{\mathrm{NB}}(t_{\mathrm{ter}}, \boldsymbol{S}) V(t_{\mathrm{ter}}, \boldsymbol{S}) \\
& \quad \left. - e^{-rt_{\mathrm{ter}}} \int_{D_{\mathrm{half}}} d\boldsymbol{S} \phi(t_{\mathrm{ter}}, \boldsymbol{S}) V(t_{\mathrm{ter}}, \boldsymbol{S}) - e^{-rt_{\mathrm{ter}}} \int_{\hat{D} \setminus D_{\mathrm{half}}} d\boldsymbol{S} \phi(t_{\mathrm{ter}}, \boldsymbol{S}) V(t_{\mathrm{ter}}, \boldsymbol{S}) \right| \\
\leq & \left| e^{-rt_{\mathrm{ter}}} \int_{D_{\mathrm{half}}} d\boldsymbol{S} \phi(t_{\mathrm{ter}}, \boldsymbol{S}) p_{\mathrm{NB}}(t_{\mathrm{ter}}, \boldsymbol{S}) V(t_{\mathrm{ter}}, \boldsymbol{S}) \right. \\
& \quad \left. - e^{-rt_{\mathrm{ter}}} \int_{D_{\mathrm{half}}} d\boldsymbol{S} \phi(t_{\mathrm{ter}}, \boldsymbol{S}) V(t_{\mathrm{ter}}, \boldsymbol{S}) \right| \\
& \quad + \left| e^{-rt_{\mathrm{ter}}} \int_{\mathbb{R}_+^d \setminus D_{\mathrm{half}}} d\boldsymbol{S} \phi(t_{\mathrm{ter}}, \boldsymbol{S}) p_{\mathrm{NB}}(t_{\mathrm{ter}}, \boldsymbol{S}) V(t_{\mathrm{ter}}, \boldsymbol{S}) \right. \\
& \quad \left. - e^{-rt_{\mathrm{ter}}} \int_{\hat{D} \setminus D_{\mathrm{half}}} d\boldsymbol{S} \phi(t_{\mathrm{ter}}, \boldsymbol{S}) V(t_{\mathrm{ter}}, \boldsymbol{S}) \right|,
\end{aligned} \tag{A82}$$

where

$$\hat{D} \coloneqq \left[\exp\left(\frac{1}{2}\left(l_1 + x_1^{(0)}\right)\right), \exp\left(\frac{1}{2}\left(x_1^{(n_{\mathrm{gr}}-1)} + u_1\right)\right)\right] \times \cdots \times \left[\exp\left(\frac{1}{2}\left(l_d + x_d^{(0)}\right)\right), \exp\left(\frac{1}{2}\left(x_d^{(n_{\mathrm{gr}}-1)} + u_d\right)\right)\right]. \tag{A83}$$

The first term in the last line in (A82) is bounded as

$$\begin{aligned}
& \left| e^{-rt_{\mathrm{ter}}} \int_{D_{\mathrm{half}}} d\boldsymbol{S}\phi(t_{\mathrm{ter}}, \boldsymbol{S}) p_{\mathrm{NB}}(t_{\mathrm{ter}}, \boldsymbol{S}) V(t_{\mathrm{ter}}, \boldsymbol{S}) - e^{-rt_{\mathrm{ter}}} \int_{D_{\mathrm{half}}} d\boldsymbol{S}\phi(t_{\mathrm{ter}}, \boldsymbol{S}) V(t_{\mathrm{ter}}, \boldsymbol{S}) \right| \\
= \; & e^{-rt_{\mathrm{ter}}} \int_{D_{\mathrm{half}}} d\boldsymbol{S}\phi(t_{\mathrm{ter}}, \boldsymbol{S})(1 - p_{\mathrm{NB}}(t_{\mathrm{ter}}, \boldsymbol{S})) V(t_{\mathrm{ter}}, \boldsymbol{S}) \\
\leq \; & \frac{e^{-rt_{\mathrm{ter}}}\epsilon}{\tilde{\tilde{A}}(d+1)} \int_{D_{\mathrm{half}}} d\boldsymbol{S}\phi_{\boldsymbol{S}}(t_{\mathrm{ter}}, \boldsymbol{S}) V(t_{\mathrm{ter}}, \boldsymbol{S}) \\
\leq \; & \frac{e^{-rt_{\mathrm{ter}}}\epsilon}{\tilde{\tilde{A}}(d+1)} \int_{\mathbb{R}_+^d} d\boldsymbol{S}\phi_{\boldsymbol{S}}(t_{\mathrm{ter}}, \boldsymbol{S}) V(t_{\mathrm{ter}}, \boldsymbol{S}) \\
\leq \; & \frac{e^{-rt_{\mathrm{ter}}}\epsilon}{\tilde{\tilde{A}}(d+1)} \int_{\mathbb{R}_+^d} d\boldsymbol{S}\phi_{\boldsymbol{S}}(t_{\mathrm{ter}}, \boldsymbol{S}) \left(\sum_{i=1}^d A_i S_i + A_0 e^{-r(T-t_{\mathrm{ter}})} \right) \\
= \; & \frac{\epsilon}{\tilde{\tilde{A}}(d+1)} \left(\sum_{i=1}^d A_i S_{i,0} + A_0 e^{-r(T-t_{\mathrm{ter}})} \right) \\
\leq \; & \epsilon,
\end{aligned} \tag{A84}$$

where we used Lemma A2.3 and (A81) at the first inequality, and (A72) at the third inequality. On the other hand, the second term of (A82) is bounded as

$$\begin{aligned}
& \left| e^{-rt_{\mathrm{ter}}} \int_{\mathbb{R}_+^d \setminus D_{\mathrm{half}}} d\boldsymbol{S}\phi(t_{\mathrm{ter}}, \boldsymbol{S}) p_{\mathrm{NB}}(t_{\mathrm{ter}}, \boldsymbol{S}) V(t_{\mathrm{ter}}, \boldsymbol{S}) - e^{-rt_{\mathrm{ter}}} \int_{\hat{D} \setminus D_{\mathrm{half}}} d\boldsymbol{S}\phi(t_{\mathrm{ter}}, \boldsymbol{S}) V(t_{\mathrm{ter}}, \boldsymbol{S}) \right| \\
= \; & \Bigg| e^{-rt_{\mathrm{ter}}} \int_{\mathbb{R}_+^d \setminus \hat{D}} d\boldsymbol{S}\phi(t_{\mathrm{ter}}, \boldsymbol{S}) p_{\mathrm{NB}}(t_{\mathrm{ter}}, \boldsymbol{S}) V(t_{\mathrm{ter}}, \boldsymbol{S}) \\
- \; & e^{-rt_{\mathrm{ter}}} \int_{\hat{D} \setminus D_{\mathrm{half}}} d\boldsymbol{S}(1 - p_{\mathrm{NB}}(t_{\mathrm{ter}}, \boldsymbol{S})) \phi(t_{\mathrm{ter}}, \boldsymbol{S}) V(t_{\mathrm{ter}}, \boldsymbol{S}) \Bigg| \\
\leq \; & e^{-rt_{\mathrm{ter}}} \int_{\mathbb{R}_+^d \setminus \hat{D}} d\boldsymbol{S}\phi(t_{\mathrm{ter}}, \boldsymbol{S}) p_{\mathrm{NB}}(t_{\mathrm{ter}}, \boldsymbol{S}) V(t_{\mathrm{ter}}, \boldsymbol{S}) \\
+ \; & e^{-rt_{\mathrm{ter}}} \int_{\hat{D} \setminus D_{\mathrm{half}}} d\boldsymbol{S}(1 - p_{\mathrm{NB}}(t_{\mathrm{ter}}, \boldsymbol{S})) \phi(t_{\mathrm{ter}}, \boldsymbol{S}) V(t_{\mathrm{ter}}, \boldsymbol{S}) \\
\leq \; & e^{-rt_{\mathrm{ter}}} \int_{\mathbb{R}_+^d \setminus D_{\mathrm{half}}} d\boldsymbol{S}\phi(t_{\mathrm{ter}}, \boldsymbol{S}) V(t_{\mathrm{ter}}, \boldsymbol{S}) \\
\leq \; & \epsilon,
\end{aligned} \tag{A85}$$

where we used Lemma A2.6 and $t_{\rm ter} < t_c$ at the last inequality. Combining these, we obtain (4.13). □

A2.3 Proof of Lemma 4.4.1

Proof. The following holds

$$\begin{aligned}\left|e^{-rT}\boldsymbol{p}\cdot\tilde{\boldsymbol{Y}}(\tau_{\rm ter})-V_0\right| &\le e^{-rT}\left|\boldsymbol{p}\cdot\left(\tilde{\boldsymbol{Y}}(\tau_{\rm ter})-\boldsymbol{Y}(\tau_{\rm ter})\right)\right| \\ &+ e^{-rT}\left|\boldsymbol{p}\cdot\boldsymbol{Y}(\tau_{\rm ter})-\int_{\tilde{\tilde{D}}}d\boldsymbol{x}\tilde{\phi}(t_{\rm ter},\boldsymbol{x})Y(\tau_{\rm ter},\boldsymbol{x})\right| \\ &+ \left|e^{-rT}\int_{\tilde{\tilde{D}}}d\boldsymbol{x}\tilde{\phi}(t_{\rm ter},\boldsymbol{x})Y(\tau_{\rm ter},\boldsymbol{x})-V_0\right|. \qquad \text{(A86)}\end{aligned}$$

The first term can be evaluated as

$$\begin{aligned}& e^{-rT}\left|\boldsymbol{p}\cdot\left(\tilde{\boldsymbol{Y}}(\tau_{\rm ter})-\boldsymbol{Y}(\tau_{\rm ter})\right)\right| \\ \le\ & e^{-rT}\|\boldsymbol{p}\|\times\left\|\tilde{\boldsymbol{Y}}(\tau_{\rm ter})-\boldsymbol{Y}(\tau_{\rm ter})\right\| \\ <\ & e^{-rT}\frac{\sqrt{\prod_{i=1}^d\Delta_i}}{(4\pi)^{d/4}\sqrt{N_{\rm gr}}(\det\rho)^{1/4}}\sqrt{N_{\rm gr}}\frac{(4\pi)^{d/4}(\det\rho)^{1/4}}{\sqrt{\prod_{i=1}^d\Delta_i}}\epsilon \\ =\ & e^{-rT}\epsilon \\ <\ & \epsilon, \qquad \text{(A87)}\end{aligned}$$

where we used (4.34), (4.37), and Lemma 2.2.1. In order to bound the second term, we note that $\boldsymbol{p}\cdot\boldsymbol{Y}(\tau_{\rm ter})$ is an approximation of $\int_{\tilde{\tilde{D}}}d\boldsymbol{x}\tilde{\phi}(t_{\rm ter},\boldsymbol{x})Y(\tau_{\rm ter},\boldsymbol{x})$ by the midpoint rule. Then, according to [89],

$$\left|\boldsymbol{p}\cdot\boldsymbol{Y}(\tau_{\rm ter})-\int_{\tilde{\tilde{D}}}d\boldsymbol{x}\tilde{\phi}(t_{\rm ter},\boldsymbol{x})Y(\tau_{\rm ter},\boldsymbol{x})\right| < \frac{1}{24}\left(\sum_{i=1}^d h_i^2\right)\times\left(\prod_{i=1}^d(u_i-l_i)\right)\times\eta \qquad \text{(A88)}$$

holds under Assumption 4.4.1, and therefore

$$e^{-rT}\left|\boldsymbol{p}\cdot\boldsymbol{Y}(\tau_{\rm ter})-\int_{\tilde{\tilde{D}}}d\boldsymbol{x}\tilde{\phi}(t_{\rm ter},\boldsymbol{x})Y(\tau_{\rm ter},\boldsymbol{x})\right| < e^{-rT}\epsilon < \epsilon \qquad \text{(A89)}$$

under (4.37). The third term can be bounded as (4.13) by Lemma 4.3.1. Combining (A87), (A89) and (4.13), we obtain the claim. □

A2.4 Proof of Lemma 4.4.2

Proof. Applying the algorithm in [4] to the ODE system (2.23) with h_i satisfying (4.37), we obtain (4.44). Since smaller h_i's lead to larger $\|F\|$,

and then larger complexity, we take as large h_i's as possible, that is,

$$h_i = \frac{u_i - l_i}{\left\lceil \frac{u_i - l_i}{\tilde{h}_i} \right\rceil} = \Theta(\tilde{h}_i) \tag{A90}$$

Then, we can evaluate the complexity by substituting s and $\|A\|$ in (4.6) with the sparsity and norm of F, respectively. The sparsity of F is $O(d^2)$, since the matrices constituting F as (2.25) have sparsity at most 4 and the total number of them is $O(d^2)$. Besides,

$$\|F\| = O\left(\max\left\{ \frac{\sqrt{\prod_{i=1}^{d} \Delta_i} d^2 \Xi \sigma_{\max}^2}{(4\pi)^{d/4} (\det \rho)^{1/4}}, d\eta \prod_{i=1}^{d} (u_i - l_i) \right\} \times \frac{d^2 \sigma_{\max}^2}{\epsilon} \right) \tag{A91}$$

for $h_i = \Theta(\tilde{h}_i)$, as we will show soon. Using these, we obtain (4.45) by simple algebra.

The remaining task is to show (A91). Note that

$$\begin{aligned} \|F\| &\leq \sum_{i=1}^{d} \frac{\sigma_i^2}{2h_i^2} \|D^{2\mathrm{nd}}\| + \sum_{i=1}^{d-1} \sum_{j=i+1}^{d} \frac{\sigma_i \sigma_j}{4h_i h_j} \|D^{1\mathrm{st}}\|^2 + \sum_{i=1}^{d} \frac{1}{2h_i} \left| r - \frac{1}{2}\sigma_i^2 \right| \|D^{1\mathrm{st}}\| \\ &= \sum_{i=1}^{d} \frac{2\sigma_i^2}{h_i^2} + \sum_{i=1}^{d-1} \sum_{j=i+1}^{d} \frac{\sigma_i \sigma_j}{h_i h_j} + \sum_{i=1}^{d} \frac{1}{h_i} \left| r - \frac{1}{2}\sigma_i^2 \right|. \end{aligned} \tag{A92}$$

Here, we used $\|D^{1\mathrm{st}}\| = 2, \|D^{2\mathrm{nd}}\| = 4$, which follows the fact that the eigenvalues of the $n \times n$ tridiagonal Toeplitz matrix

$$\begin{pmatrix} b & c & & & \\ a & b & c & & \\ & \ddots & \ddots & \ddots & \\ & & a & b & c \\ & & & a & b \end{pmatrix},$$

where $a, b, c \in \mathbb{C}$, are $b + 2\sqrt{ac}\cos\left(\frac{j\pi}{n+1}\right), j = 1, ..., n$[115]. Then, substituting h_i in (A92) by $\tilde{h}_i$, we obtain (A91) by simple algebra. □

A2.5 Proof of Theorem 4.4.1

Proof. First, we show that, for ϵ_1, ϵ_2 and ϵ_Ψ satisfying (4.61), (4.62) and (4.65), respectively, Algorithm 2 outputs ω such that (4.66). For this, we begin with writing $\left|\tilde{\Psi}_{\mathrm{mod}}\right\rangle$, the state which we obtain by applying the QLS algorithm of [23] to (4.48), in the form of

$$\left|\tilde{\Psi}_{\mathrm{mod}}\right\rangle := \frac{1}{\tilde{Z}} \left(\left|\tilde{\Psi}_{\mathrm{gar}}\right\rangle + \left|\tilde{\Psi}_1\right\rangle + \left|\tilde{\Psi}_2\right\rangle \right), \tag{A93}$$

where $\left|\tilde{\Psi}_{\rm gar}\right\rangle$, $\left|\tilde{\Psi}_1\right\rangle$ and $\left|\tilde{\Psi}_2\right\rangle$ are the unnormalized states in the forms of

$$\left|\tilde{\Psi}_{\rm gar}\right\rangle := \sum_{j=0}^{p(k+1)-1} |j\rangle\,|\psi_j\rangle\,, \left|\tilde{\Psi}_1\right\rangle := \sum_{j=p(k+1)}^{p(k+2)} |j\rangle\,|\psi_j\rangle\,, \left|\tilde{\Psi}_2\right\rangle := \sum_{j=p(k+2)}^{p(k+3)+1} |j\rangle\,|\psi_j\rangle\,, \tag{A94}$$

with some unnormalised states, respectively, and

$$\tilde{Z} := \sqrt{\left\langle\tilde{\Psi}_{\rm gar}\middle|\tilde{\Psi}_{\rm gar}\right\rangle + \langle\Psi_1|\Psi_1|\Psi_1|\Psi_1\rangle + \langle\Psi_2|\Psi_2|\Psi_2|\Psi_2\rangle} \tag{A95}$$

. Because of (4.64), we see that

$$|\left\langle\Pi|\tilde{\Psi}_{\rm mod}\middle|\Pi|\tilde{\Psi}_{\rm mod}\right\rangle - \langle\Pi|\Psi_{\rm mod}|\Pi|\Psi_{\rm mod}\rangle\,| < \epsilon_\Psi. \tag{A96}$$

Then, since E_1, the output of the step 1 in Algorithm 2, satisfies

$$|E_1 - \left\langle\tilde{\Psi}_{\rm mod}\middle|\Pi\middle|\tilde{\Psi}_{\rm mod}\right\rangle\,| < \epsilon_1, \tag{A97}$$

we obtain

$$|E_1 - \langle\Psi_{\rm mod}|\Pi|\Psi_{\rm mod}\rangle\,| < \epsilon_1 + \epsilon_\Psi. \tag{A98}$$

Similarly, since

$$\left\|\frac{1}{\tilde{Z}}\left|\tilde{\Psi}_2\right\rangle - \frac{1}{Z}\sum_{j=p(k+2)+1}^{p(k+3)+1} |j\rangle\,|\boldsymbol{\gamma}\rangle\right\| < \epsilon_\Psi \tag{A99}$$

because of (4.64) and

$$\left|E_2 - \frac{\|\left|\tilde{\Psi}_2\right\rangle\|}{\tilde{Z}}\right| < \epsilon_2, \tag{A100}$$

we obtain

$$\left|E_2 - \left\|\frac{1}{Z}\sum_{j=p(k+2)+1}^{p(k+3)+1} |j\rangle\,|\boldsymbol{\gamma}\rangle\right\|\right| = \left|E_2 - \frac{\gamma\sqrt{(p+1)N_{\rm gr}}}{Z}\right| < \epsilon_2 + \epsilon_\Psi. \tag{A101}$$

Using (A98) and (A101), we see that $\omega := e^{-rT}\gamma\sqrt{N_{\rm gr}}PE_1/E_2$ satisfies

$$\left|\omega - e^{-rT}\boldsymbol{p}\cdot\tilde{\boldsymbol{Y}}(\tau_{\rm ter})\right| < \frac{e^{-rT}PZ}{\sqrt{p+1}}(\epsilon_\Psi+\epsilon_1) + \frac{e^{-rT}(\boldsymbol{p}\cdot\tilde{\boldsymbol{Y}}(\tau_{\rm ter}))Z}{\gamma\sqrt{(p+1)N_{\rm gr}}}(\epsilon_\Psi+\epsilon_2), \tag{A102}$$

by simple algebra. Here, note that, because of (4.47) and Assumption 4.4.4,

$$Z = O\left(g\sqrt{(p+1)N_{\rm gr}}\bar{Y}(\tau_{\rm ter})\right) = O\left(g\sqrt{(p+1)N_{\rm gr}}\gamma\right) \tag{A103}$$

holds. Thus, if ϵ_1, ϵ_2 and ϵ_Ψ satisfy (4.61), (4.62) and (4.65), respectively, combining (A102), (A103) and (4.34) leads to

$$\left| \frac{e^{-rT}\gamma\sqrt{N_{\rm gr}}PE_1}{E_2} - e^{-rT}\boldsymbol{p}\cdot\tilde{\boldsymbol{Y}}(\tau_{\rm ter}) \right| = O(\epsilon). \tag{A104}$$

Finally, this and (4.38) yield (4.66).

Next, let us show that Algorithm 2 with such ϵ_1, ϵ_2 and ϵ_Ψ has the complexity (4.67). We just multiply the complexity of generating $\left|\tilde{\Psi}_{\rm mod}\right\rangle$ once, which is given by (4.45) with $\epsilon' = \epsilon_\Psi$, by the number of the generation, which is $O(\max\{1/\epsilon_1, 1/\epsilon_2\})$ since the QAE with $O(1/\delta)$ queries outputs the estimation with the error of $O(\delta)$ with a success probability higher than a specified value (say, 0.99) [6, 18]. By simple algebra, we obtain (4.67). □

A2.6 How to generate a payoff-Encoded state for a call or put option

Here, let us consider how to generate a quantum state $\left|\tilde{\boldsymbol{f}}_{\rm pay}\right\rangle/\|\tilde{\boldsymbol{f}}_{\rm pay}\|$ in which a payoff vector $\tilde{\boldsymbol{f}}_{\rm pay}$ is is amplitude encoded, for a simple case where the payoff function depends on only one asset price S_1 and takes the call-option-like form of $f_{\rm pay}(S_1) = \max\{K - S_1, 0\}$, the following discussion can be applied with a slight modification. As mentioned in Section 4.4.1, creation of such a state is possible by a procedure like Algorithm 1, if we can compute (4.43) on a quantum circuit. In the current case, this boils down to

$$f_{1,j}(b_1,\ldots,b_j) = \frac{\int_{x^L_{1,j}(b_1,\ldots,b_j)}^{\frac{1}{2}(x^L_{1,j}(b_1,\ldots,b_j)+x^R_{1,j}(b_1,\ldots,b_j))} dx_1 \left(\max\{c(e^{x_1}-K),0\}\right)^2}{\int_{x^L_{1,j}(b_1,\ldots,b_j)}^{x^R_{1,j}(b_1,\ldots,b_j)} dx_1 \left(\max\{c(e^{x_1}-K),0\}\right)^2} \tag{163}$$

where $x^L_{1,j}$ and $x^R_{1,j}$ are defined as (4.31), $j = 0, 1, \ldots, m_{\rm gr}-1$ and $b_1, \ldots, b_j \in \{0, 1\}$. When the denominator of this is 0, we regard it as 0. By straightforward calculation, we obtain $f_{1,0} = f_0^{\rm ATM}$ and, for $j \in [m_{\rm gr}-1]$, $f_{1,j}(b_1, \ldots, b_j)$ as (A105), shown at the bottom of this page.

$$f_{1,j}(b_1,\ldots,b_j)$$
$$= \begin{cases} 0, & \text{if } B_j < B_j^K \\ f_j^{\mathrm{ATM}}, & \text{if } B_j = B_j^K \\ \frac{\frac{1}{2}K^2\delta_j - 2K\left(\exp\left(\frac{1}{2}\delta_j\right)-1\right)\exp\left(x^L\right)+\frac{1}{2}\left(\exp(\delta_j)-1\right)\exp\left(2x^L\right)}{K^2\delta_j - 2K\left(\exp(\delta_j)-1\right)\exp\left(x^L\right)+\frac{1}{2}\left(\exp(2\delta_j)-1\right)\exp\left(2x^L\right)} =: f_j^{\mathrm{ITM}}(b_1,\ldots,b_j), & \text{if } B_j > B_j^K \end{cases} \tag{A105}$$

$$f_j^{\mathrm{ATM}}$$
$$:= \begin{cases} 0, & \text{if } x_K^L + \frac{1}{2}\delta_j \le k \\ \frac{K^2\left(x_K^L+\frac{1}{2}\delta_j-k\right)-2K\left(\exp\left(x_K^L+\frac{1}{2}\delta_j\right)-K\right)+\frac{1}{2}\left(\exp\left(2x_K^L+\delta_j\right)-K^2\right)}{K^2\left(x_K^L+\delta_j-k\right)-2K\left(\exp\left(x_K^L+\delta_j\right)-K\right)+\frac{1}{2}\left(\exp\left(2x_K^L+2\delta_j\right)-K^2\right)}, & \text{otherwise} \end{cases} \tag{A106}$$

Here, f_j^{ATM} is given as (A106), shown at the bottom of this page, $k := \log K$, $\delta_j := 2^{-j}(u_1 - l_1)$, x^L is the abbreviation of $x_{1,j}^L(b_1,\ldots,b_j)$, B is the positive integer with the j-th bit representation $b_1,\cdots,b_j$, B_j^K is the positive integer with the j-bit representation $\left(B_j^K\right)^{[1]}\cdots\left(B_j^K\right)^{[j]}$ satisfying $x_K^L \le k < x_K^L+\delta_j$, and $x_K^L := x_1^{(0)}$ if $j=0$ and $x_K^L := x_{1,j}^L\left(\left(B_j^K\right)^{[1]},\ldots,\left(B_j^K\right)^{[j]}\right)$ otherwise.

$f_{1,j}(b_1,\ldots,b_j)$ can be calculated by the quantum circuit in Fig. 1. This receives B_j on the top register as an input and outputs $f_j^{\mathrm{ITM}}(b_1,\ldots,b_j)$ onto the bottom register, using some ancillary registers. Here, the "$B_j = B_J^K$" gate is just a multiple controlled NOT gate and outputs 1 if $B_j = B_J^K$ and 0 otherwise, the "$B_j > B_J^K$" gate is virtually a subtracter [31] and outputs 1 if $B_j > B_j^K$ and 0 otherwise, the controlled setter of the precomputable f_j^{ATM} is a collection of $CNOT$ gates, and the controlled copier, which is made of Toffoli gates, sets $f_j^{\mathrm{ITM}}(b_1,\ldots,b_j)$ calculated by the "calc$f_j^{\mathrm{ITM}}(b_1,\ldots,b_j)$" gate to the bottom register if $B_j > B_j^K$. In this circuit, calculation of $\exp\left(x^L\right)$ in the "calc$f_j^{\mathrm{ITM}}(b_1,\ldots,b_j)$" gate, which can be done as a series of many multiplications and additions, as shown in [59] is the bottleneck part. Note that there is only one exponential calculation, since, after we get $\exp\left(x^L\right)$, we can calculate calc$f_j^{\mathrm{ITM}}(b_1,\ldots,b_j)$ by several times of four arithmetic operations [116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130] using $\exp\left(x^L\right)$ and precomputable factors such as $\frac{1}{2}\left(\exp(\delta_j)-1\right)$.

Similarly to Algorithm 2, in order to generate $\left|\tilde{\boldsymbol{f}}_{\mathrm{pay}}\right\rangle / \|\tilde{\boldsymbol{f}}_{\mathrm{pay}}\|$, we repeat calculation of $f_{1,j}(b_1,\ldots,b_j)$ and the operation

$$|\bar{0}\rangle \to \sqrt{f_{1,j}(b_1,\ldots,b_j)}|\bar{0}\rangle + \sqrt{1-f_{1,j}(b_1,\ldots,b_j)}|\bar{1}\rangle \tag{A107}$$

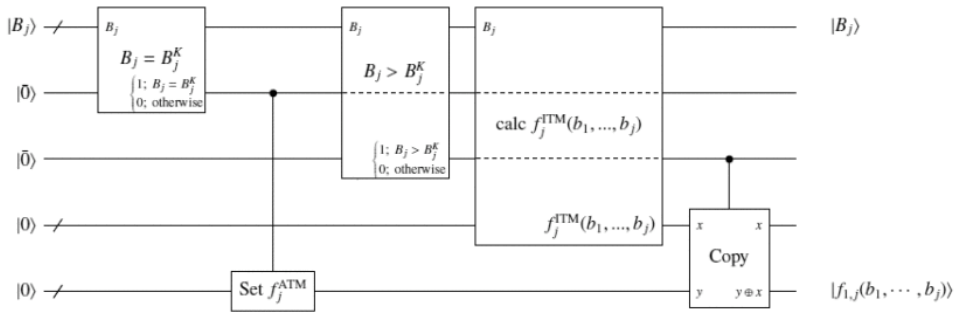


Figure 1: Quantum circuit to calculate $f_{1,j}(b_1, \ldots, b_j)$. The dashed wire going over a gate means that the corresponding register is not used in the operation of the gate. An expression in a gate and beside a wire means the input or the output of the gate.

m_{gr} times. Given $f_{1,j}(b_1, \ldots, b_j)$, we can perform (A107) by square root, arcsin, and controlled rotation. Among these, arcsin, which is done by a series of multiplications and additions similarly to exponential [31], is most costly.[2]

Consequently, the time complexity of generating $\left|\tilde{\boldsymbol{f}}_{\mathrm{pay}}\right\rangle / \|\tilde{\boldsymbol{f}}_{\mathrm{pay}}\|$ is roughly the sum of those of exponential and arcsin times m_{gr}. Let us take the number of Toffoli gates as a metric of the time complexity, as [59]. According to that paper, the Toffoli-counts for exponential and arcsin with error up to 10^{-5} are about 8000 and 5000, respectively, and we therefore estimate the Toffoli-count for generating $\left|\tilde{\boldsymbol{f}}_{\mathrm{pay}}\right\rangle / \|\tilde{\boldsymbol{f}}_{\mathrm{pay}}\|$ as $1.3 \times 10^4 \times m_{\mathrm{gr}}$.

[2]Square root can be implemented with the gate number scaling with the input qubit number similarly to multiplication [131].

A3 Pricing multi-asset derivatives by variational quantum algorithms

A3.1 Elements of the matrix and the vector of the finite difference method for the BSPDE

Here, we show the concrete elements of $D_{x_i}^{\mathrm{1st}}$ in Eqs. (5.25)(5.24), $D_{x_i}^{\mathrm{2nd}}$ in Eq. (5.24), and $\boldsymbol{C}$ in Eq. (5.21). $D_{x_i}^{\mathrm{1st}}$ and $D_{x_i}^{\mathrm{2nd}}$ are written by

$$D_{x_i}^{\mathrm{1st}} = \begin{pmatrix} 0 & x_i^{(1)} & & & & \\ -x_i^{(0)} & 0 & x_i^{(2)} & & & \\ & -x_i^{(1)} & 0 & x_i^{(3)} & & \\ & & \ddots & \ddots & \ddots & \\ & & & -x_i^{(n_{\mathrm{gr}}-3)} & 0 & x_i^{(n_{\mathrm{gr}}-1)} \\ & & & & -x_i^{(n_{\mathrm{gr}}-2)} & 0 \end{pmatrix}, \quad \text{(A108)}$$

and

$$D_{x_i}^{\mathrm{2nd}} = \begin{pmatrix} -2\left(x_i^{(0)}\right)^2 & \left(x_i^{(1)}\right)^2 & & & & \\ \left(x_i^{(0)}\right)^2 & -2\left(x_i^{(1)}\right)^2 & \left(x_i^{(2)}\right)^2 & & & \\ & \left(x_i^{(1)}\right)^2 & -2\left(x_i^{(2)}\right)^2 & \left(x_i^{(3)}\right)^2 & & \\ & & \ddots & \ddots & \ddots & \\ & & & \left(x_i^{(n_{\mathrm{gr}}-3)}\right)^2 & -2\left(x_i^{(n_{\mathrm{gr}}-2)}\right)^2 & \left(x_i^{(n_{\mathrm{gr}}-1)}\right)^2 \\ & & & & \left(x_i^{(n_{\mathrm{gr}}-2)}\right)^2 & -2\left(x_i^{(n_{\mathrm{gr}}-1)}\right)^2 \end{pmatrix}, \quad \text{(A109)}$$

respectively. $\boldsymbol{C}(\tau)$ corresponds to the boundary conditions, and its elements $C_k(\tau)$ are

$$
\begin{aligned}
C_k(\tau) = &\sum_{i=1}^{d} \frac{\sigma_i^2}{2h_i^2} \left[(l_i + h_i)^2 \delta_{k_i,0} \bar{V}_i^{\mathrm{LB}}(\tau, \boldsymbol{x}_{\wedge i}^{(k)}) + (l_i + n_{\mathrm{gr}} h_i)^2 \delta_{k_i, n_{\mathrm{gr}}-1} \bar{V}_i^{\mathrm{UB}}(\tau, \boldsymbol{x}_{\wedge i}^{(k)}) \right] \\
&+ \sum_{i=1}^{d-1} \sum_{j=i+1}^{d} \frac{\sigma_i \sigma_j \rho_{ij}}{4 h_i h_j} \\
&\times \Big[-(l_i + h_i)(l_j + (k_j+1)h_j) \delta_{k_i,0} \bar{V}_i^{\mathrm{LB}}(\tau, \boldsymbol{x}_{\wedge i}^{(k)}) \\
&-(l_i + (k_i+1)h_i)(l_j + h_j) \delta_{k_j,0} \bar{V}_j^{\mathrm{LB}}(\tau, \boldsymbol{x}_{\wedge j}^{(k)}) \\
&+ (l_i + n_{\mathrm{gr}} h_i)(l_j + (k_j+1)h_j) \delta_{k_i, n_{\mathrm{gr}}-1} \bar{V}_i^{\mathrm{UB}}(\tau, \boldsymbol{x}_{\wedge i}^{(k)}) \\
&+(l_i + (k_i+1)h_i)(l_j + n_{\mathrm{gr}} h_j) \delta_{k_j, n_{\mathrm{gr}}-1} \bar{V}_j^{\mathrm{UB}}(\tau, \boldsymbol{x}_{\wedge j}^{(k)}) \Big] \\
&+ r \sum_{i=1}^{d} \frac{1}{2h_i} \left[(l_i + n_{\mathrm{gr}} h_i) \delta_{k_i, n_{\mathrm{gr}}-1} \bar{V}_i^{\mathrm{UB}}(\tau, \boldsymbol{x}_{\wedge i}^{(k)}) - (l_i + h_i) \delta_{k_i,0} \bar{V}_i^{\mathrm{LB}}(\tau, \boldsymbol{x}_{\wedge i}^{(k)}) \right],
\end{aligned}
\tag{A110}
$$

where $\bar{V}_i^{\mathrm{UB}}(\tau, \boldsymbol{x}_{\wedge i}^{(k)}) = V_i^{\mathrm{UB}}(\tau, \boldsymbol{x}_{\wedge i}^{(k)})$ and $\bar{V}_i^{\mathrm{LB}}(\tau, \boldsymbol{x}_{\wedge i}^{(k)}) = V_i^{\mathrm{LB}}(\tau, \boldsymbol{x}_{\wedge i}^{(k)})$.

A3.2 Decomposition of matrices

As discussed in Sec. 5.3, we need to express F and $|\boldsymbol{C}(\tau)\rangle$ in terms of linear combination of quantum gates to perform the VQS for the BSPDE. Here, we show that such decomposition is possible. The decomposition of F is based on the way shown in Refs. [70, 30]. We also obtain a linear combination of quantum gates that generates $|\boldsymbol{C}(\tau)\rangle$ by slightly modifying the decomposition of F. For simplicity, we assume $N_{\mathrm{gr}} = 2^n$ where n is the number of qubits. $D_{x_i}^{\mathrm{1st}}, D_{x_i}^{\mathrm{2nd}}$ in Eqs. (5.25)(5.24) are decomposed as follows,

$$
\begin{aligned}
D_{x_i}^{\mathrm{1st}} = &\, l_i D^{\mathrm{1st}} + h_i \left(\mathrm{Dec}(n)(J(n) + 2I^{\otimes n}) \right. \\
&\left. - \mathrm{Inc}(n) \left(J(n) + I^{\otimes n} \right) \right),
\end{aligned}
\tag{A111}
$$

$$
\begin{aligned}
D_{x_i}^{\mathrm{2nd}} = &\, l_i^2 D^{\mathrm{2nd}} \\
&+ 2 l_i h_i \left(\mathrm{Dec}(n) - 2I^{\otimes n} + \mathrm{Inc}(n) \right) (J(n) + I) \\
&+ h_i^2 \left(\mathrm{Dec}(n) - 2I^{\otimes n} + \mathrm{Inc}(n) \right) (J(n) + I)^2.
\end{aligned}
\tag{A112}
$$

Here, we define

$$D^{\mathrm{1st}} \coloneqq -\mathrm{Inc}(n) + \mathrm{Dec}(n) \tag{A113}$$

$$D^{\mathrm{2nd}} \coloneqq \mathrm{Inc}(n) + \mathrm{Dec}(n) - 2I^{\otimes n} \tag{A114}$$

$$J(n) \coloneqq \sum_{i=0}^{2^n-1} i\,|i\rangle\langle i| = \frac{2^n-1}{2} I^{\otimes n} - \sum_{i=1}^{n} 2^{n-i-1} Z_i \tag{A115}$$

$$\mathrm{Inc}(n) \coloneqq \sum_{i=0}^{2^n-2} |i+1\rangle\langle i| \tag{A116}$$

$$\mathrm{Dec}(n) \coloneqq \sum_{i=1}^{2^n-1} |i-1\rangle\langle i| \tag{A117}$$

$$\tag{A118}$$

where $Z_i \coloneqq I^{\otimes i-1} \otimes Z \otimes I^{\otimes n-i}$. $\mathrm{Inc}(n), \mathrm{Dec}(n)$ are constructed by following operators

$$\mathrm{CycInc}(n) \coloneqq \sum_{i=0}^{2^n-1} |i+1\rangle\langle i|\,, \tag{A119}$$

$$\mathrm{CycDec}(n) \coloneqq \sum_{i=1}^{2^n-1} |i-1\rangle\langle i|\,, \tag{A120}$$

$$\tag{A121}$$

where we define $|-1\rangle \coloneqq |2^n-1\rangle, |2^n\rangle \coloneqq |0\rangle$. $\mathrm{CycInc}(n), \mathrm{CycDec}(n)$ can be decomposed into a product of $O(n)$ Toffoli, CNOT, X gates with $O(n)$ ancilla qubits [132]. With these circuits, we obtain

$$\mathrm{Inc}(n) = \frac{1}{2}\mathrm{CycInc}(n)(C^{n-1}Z + I^{\otimes n}), \tag{A122}$$

$$\mathrm{Dec}(n) = \frac{1}{2}(C^{n-1}Z + I^{\otimes n})\mathrm{CycDec}(n). \tag{A123}$$

$C^{n-1}Z \coloneqq \sum_{i=0}^{2^n-2} |i\rangle\langle i| - |2^n-1\rangle\langle 2^n-1|$ is an n qubit control Z gate and can be implemented as a product of $O(n^2)$ Toffoli, CNOT, and single-qubit gates [41]. We can express $D_{x_i}^{\mathrm{1st}}$ and $D_{x_i}^{\mathrm{2nd}}$ as sums of $O(n^2)$ unitary operators, each of which is a product of $O(n^2)$ few-qubit gates. Then, the first term of Eq. (5.24) is a sum of $O(dn^2)$ operators, each of which is made by $O(n^2)$ few-qubit gates. The second term is the sum of $O(d^2n^4)$ unitary operators each of which is made by $O(n^2)$ few-qubit gates. From Eqs. (5.24) and (5.25), we see that F can eventually be expressed as a sum of $O(d^2n^4)$ unitary operators each of which is made by $O(n^2)$ few-qubit gates.

It is also necessary to construct a linear combination of unitary operators that outputs the quantum state $|\boldsymbol{C}(\tau)\rangle = \sum_{k=1}^{N_{\mathrm{gr}}} C_k(\tau)\,|k\rangle$. Here, we consider

specific cases where $f_{\text{pay}}(\boldsymbol{S}(T)) = \max(a_0 + \sum_{i=1}^{d} a_j S_j(T) - K, 0)$, and some assets have knock-out conditions. These are the cases where the typical boundary conditions introduced in Sec. 5.2.2 are compounded. In these cases, we can write

$$|\boldsymbol{C}(\tau)\rangle = \tilde{G}\,|0\rangle = 2^{nd/2} G(\tau) H^{\otimes nd}\,|0\rangle \tag{A124}$$

where

$$\begin{aligned}
G(\tau) = &\sum_{i=1}^{d} \frac{\sigma_i^2}{2h_i}\left[(l_i+h_i)^2 G_i^{(0)} B_i^{\text{LB}}(\tau)\delta_i^{\text{UB}} + (l_i+n_{\text{gr}}h_i)^2 G_i^{(n_{\text{gr}}-1)} B_i^{\text{UB}}(\tau)\delta_i^{\text{UB}}\right] \\
&+ \sum_{i=1}^{d-1}\sum_{j=i+1}^{d} \frac{\sigma_i\sigma_j\rho_{ij}}{4h_ih_j} \\
&\times \Big[-(l_i+h_i)(l_j I^{\otimes dn} + h_j J_j) G_i^{(0)} B_i^{\text{LB}}(\tau)\delta_i^{\text{LB}} \\
&-(l_i I^{\otimes dn} + h_i J_i(n))(l_j+h_j) G_j^{(0)} B_j^{\text{LB}}(\tau)\delta_i^{\text{LB}} \\
&+ (l_i+n_{\text{gr}}h_i)(l_j I^{\otimes dn} + h_j J_j) G_i^{(n_{\text{gr}}-1)} B_i^{\text{UB}}(\tau)\delta_i^{\text{UB}} \\
&+(l_i I^{\otimes dn} + h_i J_i(n))(l_j+n_{\text{gr}}h_j) G_j^{(n_{\text{gr}}-1)} B_j^{\text{UB}}(\tau)\delta_i^{\text{UB}}\Big] \\
&+ r\sum_{i=1}^{d}\frac{1}{2h_i}\left[(l_i+n_{\text{gr}}h_i) G_i^{(n_{\text{gr}}-1)} B_i^{\text{UB}}(\tau)\delta_i^{\text{LB}} - (l_i+h_i) G_i^{(0)} B_i^{\text{LB}}(\tau)\delta_i^{LB}\right],
\end{aligned} \tag{A125}$$

where

$$\delta_i^{\text{UB}} = \begin{cases} 0 & \textit{up and out barrier} \text{ is set the } i\text{-th asset} \\ 1 & \text{otherwise} \end{cases}, \tag{A126}$$

$$\delta_i^{\text{LB}} = \begin{cases} 0 & \textit{down and out barrier} \text{ is set to the } i\text{-th asset} \\ 1 & \text{otherwise} \end{cases}, \tag{A127}$$

and

$$G_i^{(0)} = I^{\otimes n(i-1)} \otimes |0\rangle\langle 0|^{\otimes n} \otimes I^{n(d-i)} \tag{A128}$$

$$G_i^{(n_{\rm gr}-1)} = I^{\otimes n(i-1)} \otimes |1\rangle\langle 1|^{\otimes n} \otimes I^{n(d-i)} \tag{A129}$$

$$\begin{aligned} B_i^{\rm UB}(\tau) &= e^{-r\tau} a_0 I^{\otimes nd} \\ &+ \sum_{1\le j\le d, j\neq i} a_j \left(l_j I^{\otimes nd} + (n_{\rm gr}-1) h_j J_j + I^{\otimes nd} \right) \\ &+ a_i l_i I^{\otimes nd} \end{aligned} \tag{A130}$$

$$\begin{aligned} B_i^{\rm LB}(\tau) &= e^{-r\tau} a_0 I^{\otimes nd} \\ &+ \sum_{1\le j\le d, j\neq i} a_j \left(l_j I^{\otimes nd} + (n_{\rm gr}-1) h_j J_j + I^{\otimes nd} \right) \\ &+ a_i u_i I^{\otimes nd} \end{aligned} \tag{A131}$$

$$J_i(n) = I^{\otimes n(i-1)} \otimes (J(n) + I^{\otimes n}) \otimes I^{\otimes n(d-i)} \tag{A132}$$

where $|0\rangle\langle 0|^{\otimes n} = \frac{1}{2}\left(I^{\otimes n} - X^{\otimes n} \cdot C^n Z \cdot X^{\otimes n}\right)$ and $|1\rangle\langle 1|^{\otimes n} = \frac{1}{2}\left(I^{\otimes n} + C^{n-1}Z\right)$. $G_i^{(0)}$ and $G_i^{(n_{\rm gr})}$ are expressed as a sum of $O(1)$ unitary operator each of which is made by $O(n^2)$ few-qubit gates. $B_i^{\rm UB}$ and $B_i^{\rm LB}$ are expressed as a sum of $O(dn)$ unitary operators, each of which is made by $O(n)$ few-qubit gates. Thus, $G(\tau)$ is a sum of $O(d^2 \times n \times dn) = O(d^3 n^2)$ unitary operators each of which is made by $O(n^2)$ few-qubit gates.

A3.3 Variational principle for VQS

Here, we derive Eq. (2.54) from a variational principle. The square of the difference between both sides of Eq. (2.50) is

$$
\begin{aligned}
&\left\| \frac{d}{dt} |\tilde{v}(\boldsymbol{\theta}(t))\rangle - L(t) |\tilde{v}(\boldsymbol{\theta}(t))\rangle - |\boldsymbol{u}(t)\rangle \right\|^2 \\
&= \left\| \frac{d}{dt} |\tilde{v}(\boldsymbol{\theta}(t))\rangle - L(t) |\tilde{v}(\boldsymbol{\theta}(t))\rangle \right\|^2 \\
&- 2\,\mathrm{Re}\left[\langle \boldsymbol{u}(t)| \left(\frac{d}{dt} |\tilde{v}(\boldsymbol{\theta}(t))\rangle - L(t) |\tilde{v}(\boldsymbol{\theta}(t))\rangle \right) \right] + \| |\boldsymbol{u}(t)\rangle \|^2 \\
&= \left\| \frac{d}{dt} |\tilde{v}(\boldsymbol{\theta}(t))\rangle \right\|^2 - 2\,\mathrm{Re}\left[\langle \tilde{v}(\boldsymbol{\theta}(t))| L(t) \frac{d}{dt} |\tilde{v}(\boldsymbol{\theta}(t))\rangle \right] \\
&+ \| L(t) |\tilde{v}(\boldsymbol{\theta}(t))\rangle \|^2 \\
&- 2\,\mathrm{Re}\left[\langle \boldsymbol{u}(t)| \left(\frac{d}{dt} |\tilde{v}(\boldsymbol{\theta}(t))\rangle - L(t) |\tilde{v}(\boldsymbol{\theta}(t))\rangle \right) \right] + \| |\boldsymbol{u}(t)\rangle \|^2 \\
&= 2\,\mathrm{Re} \sum_{j,k} \frac{d \langle \tilde{v}(\theta_j(t))|}{d\theta_j(t)} \frac{d |\tilde{v}(\theta_k(t))\rangle}{d\theta_k(t)} \dot{\theta}_j \dot{\theta}_k \\
&- 2\,\mathrm{Re}\left[\frac{d \langle \tilde{v}(\boldsymbol{\theta}(t))|}{d\theta_j(t)} L(t) |\tilde{v}(\boldsymbol{\theta}(t))\rangle + \frac{d \langle \tilde{v}(\boldsymbol{\theta}(t))|}{d\theta_j(t)} |\boldsymbol{u}(t)\rangle \right] \dot{\theta}_j \\
&+ 2\,\mathrm{Re}\left[\langle \boldsymbol{u}(t)| L(t) |\tilde{v}(\boldsymbol{\theta}(t))\rangle \right] + \| L(t) |\tilde{v}(\boldsymbol{\theta}(t))\rangle \|^2 + \| |\boldsymbol{u}(t)\rangle \|^2 . \qquad \text{(A133)}
\end{aligned}
$$

Then, the first order variation of the r.h.s. of Eq. A133 is

$$
\begin{aligned}
&\delta \left\| \frac{d}{dt} |\tilde{v}(\boldsymbol{\theta}(t))\rangle - L(t) |\tilde{v}(\boldsymbol{\theta}(t))\rangle - |\boldsymbol{u}(t)\rangle \right\|^2 \\
&= 2\,\mathrm{Re} \sum_{j,k} \frac{d \langle \tilde{v}(\theta_j(t))|}{d\theta_j(t)} \frac{d |\tilde{v}(\theta_k(t))\rangle}{d\theta_k(t)} \dot{\theta}_j \delta\dot{\theta}_k \\
&- 2\,\mathrm{Re}\left[\frac{d \langle \tilde{v}(\boldsymbol{\theta}(t))|}{d\theta_j(t)} L(t) |\tilde{v}(\boldsymbol{\theta}(t))\rangle + \frac{d \langle \tilde{v}(\boldsymbol{\theta}(t))|}{d\theta_j(t)} |\boldsymbol{u}(t)\rangle \right] \delta\dot{\theta}_j . \qquad \text{(A134)}
\end{aligned}
$$

Thus, we obtain Eq. (2.54).

A3.4 Quantum circuits to evaluate $\mathcal{M}_{i,j}$ and $\mathcal{V}_i$

Here, we show the quantum circuits to evaluate $\mathcal{M}_{i,j}$ and $\mathcal{V}_j$. Without loss of generality, we can set $i \leq j$. The terms in Eqs. (2.55) and (2.56) are

written by

$$\mathrm{Re}\left(\frac{\partial\langle\tilde{v}(\boldsymbol{\theta}(t))|}{\partial\theta_i}\frac{\partial|\tilde{v}(\boldsymbol{\theta}(t))\rangle}{\partial\theta_j}\right)=\begin{cases}\theta_0(t)^2\,\mathrm{Re}\left(\langle\boldsymbol{v}_0|\,R_1^\dagger\cdots R_{i-1}^\dagger G_m^\dagger R_i^\dagger \right.\\ \qquad\left.\cdots R_{j-1}^\dagger G_j R_{j-1}\cdots R_1\,|\boldsymbol{v}_0\rangle\right)\\ \text{for } 0<i\le j\le N_a & \text{(A135)}\\ \theta_0(t)\,\mathrm{Re}\left(\langle\boldsymbol{v}_0|\,R_1^\dagger\cdots R_{j-1}^\dagger G_j^\dagger R_{j-1}\cdots R_1\,|\boldsymbol{v}_0\rangle\right)\\ \text{for } 0=i<j\le N_a & \text{(A136)}\\ 1\\ \text{for } m=n=0 & \text{(A137)}\end{cases}$$

$$\mathrm{Re}\left(\frac{\partial\langle\tilde{v}(\boldsymbol{\theta}(t))|}{\partial\theta_i}U_k^L\,|\tilde{v}(\boldsymbol{\theta}(t)\rangle\right)=\begin{cases}\theta_0(t)\,\mathrm{Re}\left(\langle\boldsymbol{v}_0|\,R_1^\dagger\cdots R_{i-1}^\dagger G_i^\dagger R_i^\dagger \right.\\ \qquad\left.\cdots R_{N_a}^\dagger U_k^L R_{N_a}\cdots R_1\,|\boldsymbol{v}_0\rangle\right)\\ \text{for } i\neq 0 & \text{(A138)}\\ \mathrm{Re}\left(\langle\boldsymbol{v}_0|\,R_1^\dagger\cdots R_{N_a}^\dagger U_k^{L\dagger} R_{N_a}\cdots R_1\,|\boldsymbol{v}_0\rangle\right)\\ \text{for } i=0 & \text{(A139)}\end{cases}$$

$$\mathrm{Re}\left(\frac{\partial\langle\tilde{v}(\boldsymbol{\theta}(t))|}{\partial\theta_m}U_l^u\,|0\rangle\right)=\begin{cases}\theta_0(t)\,\mathrm{Re}\left(\langle\boldsymbol{v}_0|\,R_1^\dagger\cdots R_{i-1}^\dagger G_m^\dagger R_i^\dagger\cdots R_{N_a}^\dagger U_l^u\,|0\rangle\right)\\ \text{for } i\neq 0 & \text{(A140)}\\ \mathrm{Re}\left(\langle\boldsymbol{v}_0|\,R_1^\dagger\cdots R_{N_a}^\dagger U_l^{u\dagger}\,|0\rangle\right)\\ \text{for } i=0 & \text{(A141)}\end{cases}$$

We can evaluate these terms using quantum circuits depicted in Fig. 2. Note that, although the quantum circuit evaluating Eqs. (A140) and (A141) contains the control-$\mathcal{R}U_v$ gate, where

$$\mathcal{R}=R_1\cdots R_{m-1}G_mR_m\cdots R_{N_a} \tag{A142}$$

for Eq. (A140) and $R_1\cdots R_{i-1}G_iR_i\cdots R_{N_a}$ for Eq. (A141) respectively, in the case where all boundary conditions are knock-out barriers, that is, in the case of $|\boldsymbol{C}(t)\rangle=\boldsymbol{0}$, we do not need to evaluate Eqs. (A140) and (A141).

A3.5 Lower bound of Ξ

Here, we evaluate the lower bound of Ξ and show that the Ξ does not decrease exponentially with respect to the number of assets d. Using the inequality

$$\min(a^2,b^2)\le\frac{1}{4}(a+b)^2 \tag{A143}$$

for $a, b \in \mathbb{R}_+$, we obtain

$$\begin{aligned} t_{\mathrm{ter}} &\leq \min\left\{ \frac{2\left(\log \frac{u_i}{s_{i,0}}\right)^2}{25\sigma_i^2 \log \frac{2\tilde{A}d(d+1)}{\epsilon}}, \frac{2\left(\log \frac{s_{i,0}}{l_i}\right)^2}{25\sigma_i^2 \log \frac{2\tilde{A}d(d+1)}{\epsilon}} \right\} \\ &\leq \frac{\left(\log \frac{u_i}{l_i}\right)^2}{50\sigma_i^2 \log \frac{2\tilde{A}d(d+1)}{\epsilon}}, \end{aligned} \tag{A144}$$

for $i \in [d]$. From Eqs. (5.44)(A144), Ξ is evaluated by

$$\Xi \geq \zeta B^2 \left(\frac{25}{4\pi} \log \frac{2\tilde{A}d(d+1)}{\epsilon}\right)^{d/2} \prod_{i=1}^{d} \frac{\frac{u_i}{l_i} - 1}{\log \frac{u_i}{l_i}} \tag{A145}$$

As easily verified by elementary analysis, for any $z > 1$,

$$\frac{z-1}{\log z} \geq 1 \tag{A146}$$

holds, and then, we obtain

$$\Xi \geq \zeta B^2 \left(\frac{25}{4\pi} \log \frac{2\tilde{A}d(d+1)}{\epsilon}\right)^{d/2}. \tag{A147}$$

Thus, we can see that Ξ does not decrease exponentially with respect to d.

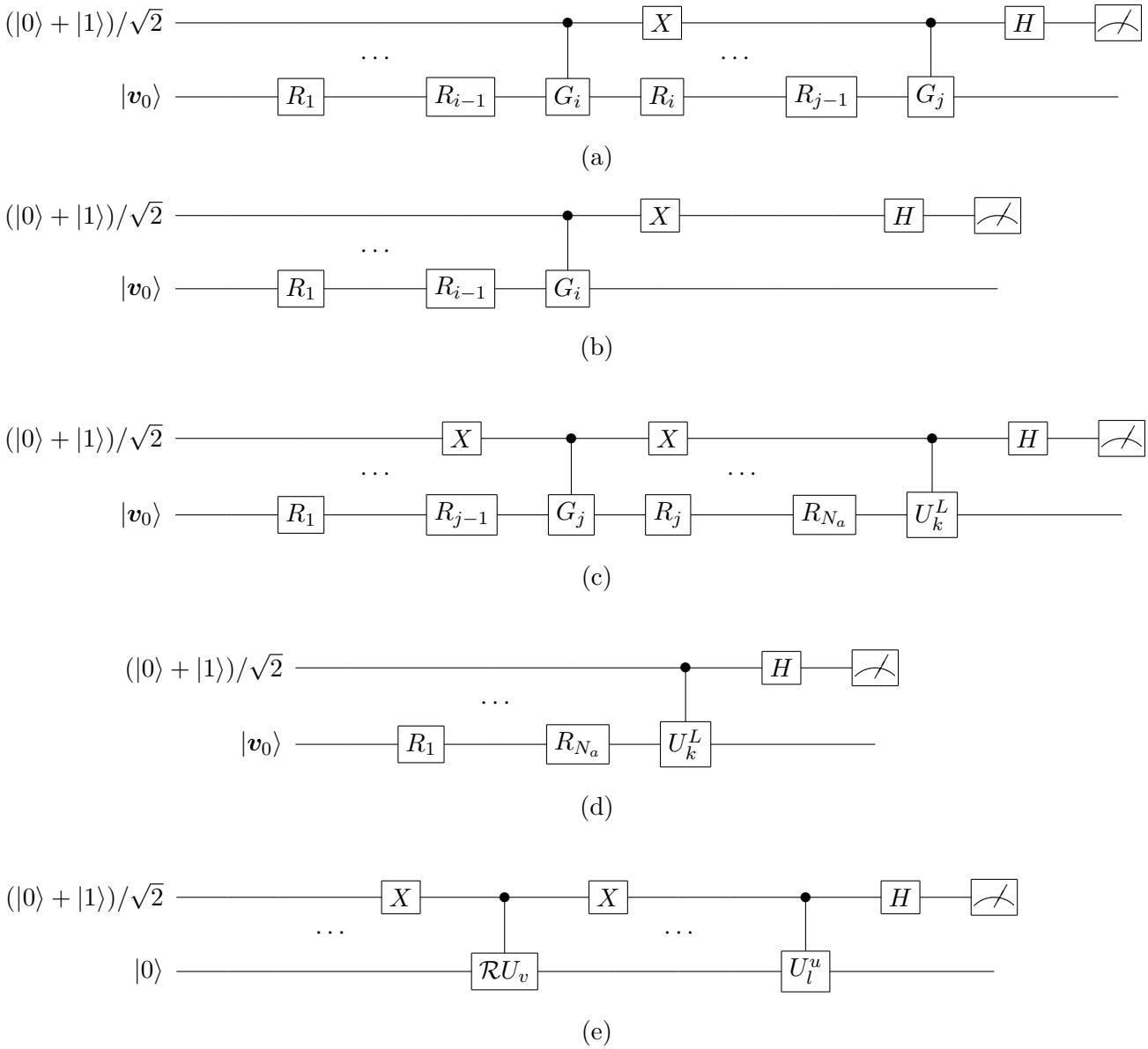


Figure 2: Quantum circuits for evaluating (a) Eq. (A135), (b) Eq. (A136), (c) Eq. (A138), (d) Eq. (A138), and (e) Eqs. (A140) and (A141). $\mathcal{R}$ is $R_1 \cdots R_{i-1} G_i R_i \cdots R_{N_a}$ for Eq. (A140) and $R_1 \cdots R_{i-1} G_i R_i \cdots R_{N_a}$ for Eq. (A141) respectively [5].

Bibliography

[1] John C. Hull. *Options, Futures, and Other Derivatives, 9th edition.* Pearson, 2014.

[2] Steven E. Shreve. *Stochastic Calculus for Finance II.* Number 11 in Springer Finance Textbooks. Springer, 2004.

[3] P.W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proc. 35th Annu. Symp. Found. Comput. Sci.*, pages 124–134. IEEE Comput. Soc. Press, nov 2002.

[4] Dominic W. Berry, Andrew M. Childs, Aaron Ostrander, and Guoming Wang. Quantum Algorithm for Linear Differential Equations with Exponentially Improved Dependence on Precision. *Communications in Mathematical Physics*, 356(3):1057–1081, 2017.

[5] Suguru Endo, Jinzhao Sun, Ying Li, Simon C. Benjamin, and Xiao Yuan. Variational Quantum Simulation of General Processes. *Physical Review Letters*, 125(1):010501, 2020.

[6] Gilles Brassard, Peter Høyer, Michele Mosca, and Alain Tapp. *Quantum amplitude amplification and estimation.* Contemporary Mathematics, 2002.

[7] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. Quantum machine learning. *Nature*, 549(7671):195–202, 2017.

[8] John Preskill. Quantum Computing in the NISQ era and beyond. *Quantum*, 2:79, August 2018.

[9] M. Cerezo, Andrew Arrasmith, Ryan Babbush, Simon C. Benjamin, Suguru Endo, Keisuke Fujii, Jarrod R. McClean, Kosuke Mitarai, Xiao Yuan, Lukasz Cincio, and Patrick J. Coles. Variational quantum algorithms. *Nature Reviews Physics*, 3(9):625–644, 2021.

[10] Martin Plesch and Časlav Brukner. Quantum-state preparation with universal gate decompositions. *Phys. Rev. A*, 83:032302, Mar 2011.

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

[12] Seth Lloyd and Christian Weedbrook. Quantum generative adversarial learning. *Physical review letters*, 121(4):040502, 2018.

[13] Pierre-Luc Dallaire-Demers and Nathan Killoran. Quantum generative adversarial networks. *Physical Review A*, 98(1):012324, 2018.

[14] Haozhen Situ, Zhimin He, Yuyi Wang, Lvzhou Li, and Shenggen Zheng. Quantum generative adversarial network for generating discrete distribution. *Information Sciences*, 538:193–208, 2020.

[15] Christa Zoufal, Aurélien Lucchi, and Stefan Woerner. Quantum generative adversarial networks for learning and loading random distributions. *npj Quantum Information*, 5(1):1–9, 2019.

[16] Jin-Guo Liu and Lei Wang. Differentiable learning of quantum circuit born machines. *Physical Review A*, 98(6):062324, 2018.

[17] Oleksandr Kyriienko, Annie E Paine, and Vincent E Elfving. Protocols for trainable and differentiable quantum generative modelling. *arXiv preprint arXiv:2202.08253*, 2022.

[18] Ashley Montanaro. Quantum speedup of Monte Carlo methods. *Proceedings. Mathematical, Physical, and Engineering Sciences / The Royal Society*, 471(2181):20150301, 2015.

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

[20] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. *Phys. Rev. Lett.*, 103(15):150502, oct 2009.

[21] Andris Ambainis. Variable time amplitude amplification and quantum algorithms for linear algebra problems. In Thomas Wilke Christoph Dürr, editor, *STACS'12 (29th Symposium on Theoretical Aspects of Computer Science)*, volume 14, pages 636–647, Paris, France, February 2012. LIPIcs.

[22] B. D. Clader, B. C. Jacobs, and C. R. Sprouse. Preconditioned quantum linear system algorithm. *Phys. Rev. Lett.*, 110:250504, Jun 2013.

[23] Andrew M. Childs, Robin Kothari, and Rolando D. Somma. Quantum algorithm for systems of linear equations with exponentially improved

dependence on precision. *SIAM Journal on Computing*, 46(6):1920–1950, 2017.

[24] Xiao Yuan, Suguru Endo, Qi Zhao, Ying Li, and Simon C. Benjamin. Theory of variational quantum simulation. *Quantum*, 3:191, October 2019.

[25] A.D. McLachlan. A variational solution of the time-dependent schrodinger equation. *Molecular Physics*, 8(1):39–44, 1964.

[26] Arthur G. Rattew and Bálint Koczor. Preparing arbitrary continuous functions in quantum registers with logarithmic complexity, 2022.

[27] Javier Gonzalez-Conde, Ángel Rodríguez-Rozas, Enrique Solano, and Mikel Sanz. Pricing financial derivatives with exponential quantum speedup, 2021.

[28] Santosh Kumar Radha. Quantum option pricing using wick rotated imaginary time evolution, 2021.

[29] Filipe Fontanela, Antoine Jacquier, and Mugad Oumgari. Short communication: A quantum algorithm for linear pdes arising in finance. *SIAM Journal on Financial Mathematics*, 12(4):SC98–SC114, 2021.

[30] Hedayat Alghassi, Amol Deshmukh, Noelle Ibrahim, Nicolas Robles, Stefan Woerner, and Christa Zoufal. A variational quantum algorithm for the feynman-kac formula. *Quantum*, 6:730, 2022.

[31] Kazuya Kaneko, Koichi Miyamoto, Naoyuki Takeda, and Kazuyoshi Yoshino. Quantum pricing with a smile: implementation of local volatility model on quantum computer. *EPJ Quantum Technology*, 9(1):7, 2022.

[32] N. G. Van Kampen. *Stochastic Processes in Physics and Chemistry*. North Holland, 2011.

[33] Darren J. Wilkinson. Stochastic modelling for quantitative description of heterogeneous biological systems. *Nat. Rev. Genet.*, 10(2):122–133, 2009.

[34] Nicholas Metropolis and S. Ulam. The Monte Carlo Method. *J. Am. Stat. Assoc.*, 44(247):335–341, 1949.

[35] Fischer Black and Myron Scholes. The pricing of options and corporate liabilities. *J. Polit. Econ.*, 81(3):637–657, may 1973.

[36] Francis A. Longstaff and Eduardo S. Schwartz. Valuing American Options by Simulation: A Simple Least-Squares Approach. *The Review of Financial Studies*, 14(1):113–147, 2001.

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

[38] John Preskill. Quantum Computing in the NISQ era and beyond. *Quantum*, 2:79, jan 2018.

[39] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C. Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando G. S. L. Brandao, David A. Buell, Brian Burkett, Yu Chen, Zijun Chen, Ben Chiaro, Roberto Collins, William Courtney, Andrew Dunsworth, Edward Farhi, Brooks Foxen, Austin Fowler, Craig Gidney, Marissa Giustina, Rob Graff, Keith Guerin, Steve Habegger, Matthew P. Harrigan, Michael J. Hartmann, Alan Ho, Markus Hoffmann, Trent Huang, Travis S. Humble, Sergei V. Isakov, Evan Jeffrey, Zhang Jiang, Dvir Kafri, Kostyantyn Kechedzhi, Julian Kelly, Paul V. Klimov, Sergey Knysh, Alexander Korotkov, Fedor Kostritsa, David Landhuis, Mike Lindmark, Erik Lucero, Dmitry Lyakh, Salvatore Mandrà, Jarrod R. McClean, Matthew McEwen, Anthony Megrant, Xiao Mi, Kristel Michielsen, Masoud Mohseni, Josh Mutus, Ofer Naaman, Matthew Neeley, Charles Neill, Murphy Yuezhen Niu, Eric Ostby, Andre Petukhov, John C. Platt, Chris Quintana, Eleanor G. Rieffel, Pedram Roushan, Nicholas C. Rubin, Daniel Sank, Kevin J. Satzinger, Vadim Smelyanskiy, Kevin J. Sung, Matthew D. Trevithick, Amit Vainsencher, Benjamin Villalonga, Theodore White, Z. Jamie Yao, Ping Yeh, Adam Zalcman, Hartmut Neven, and John M. Martinis. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510, Oct 2019.

[40] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, STOC ' 96, page 212–219, New York, NY, USA, 1996. Association for Computing Machinery.

[41] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010.

[42] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J. Love, Alán Aspuru-Guzik, and Jeremy L. O'Brien. A variational eigenvalue solver on a quantum processor. apr 2013.

[43] Abhinav Kandala, Antonio Mezzacapo, Kristan Temme, Maika Takita, Markus Brink, Jerry M. Chow, and Jay M. Gambetta. Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. *Nature*, 549(7671):242–246, sep 2017.

[44] Sam McArdle, Suguru Endo, Alán Aspuru-Guzik, Simon C. Benjamin, and Xiao Yuan. Quantum computational chemistry. *Rev. Mod. Phys.*, 92(1):15003, 2020.

[45] Yudong Cao, Jonathan Romero, Jonathan P. Olson, Matthias Degroote, Peter D. Johnson, Mária Kieferová, Ian D. Kivlichan, Tim Menke, Borja Peropadre, Nicolas P. D. Sawaya, Sukin Sim, Libor Veis, and Alán Aspuru-Guzik. Quantum chemistry in the age of quantum computing. *Chemical Reviews*, 119(19):10856–10915, Oct 2019.

[46] Kosuke Mitarai, Makoto Negoro, Masahiro Kitagawa, and Keisuke Fujii. Quantum circuit learning. *Physical Review A*, 98(3):032309, 2018.

[47] Maria Schuld and Nathan Killoran. Quantum Machine Learning in Feature Hilbert Spaces. *Phys. Rev. Lett.*, 122(4):40504, 2019.

[48] Edward Farhi and Hartmut Neven. Classification with Quantum Neural Networks on Near Term Processors. feb 2018.

[49] Iris Cong, Soonwon Choi, and Mikhail D Lukin. Quantum convolutional neural networks. *Nat. Phys.*, 15(12):1273–1278, 2019.

[50] P. P. Boyle. Option valuation using a three-jump process. *Int. Options J.*, 3:7–12, 1986.

[51] Ying Li and Simon C. Benjamin. Efficient variational quantum simulator incorporating active error minimization. *Phys. Rev. X*, 7:021050, Jun 2017.

[52] Sam McArdle, Tyson Jones, Suguru Endo, Ying Li, Simon C. Benjamin, and Xiao Yuan. Variational ansatz-based quantum simulation of imaginary time evolution. *npj Quantum Inf.*, 5(1):1–15, 2019.

[53] Suguru Endo, Zhenyu Cai, Simon C. Benjamin, and Xiao Yuan. Hybrid quantum-classical algorithms and quantum error mitigation. *Journal of the Physical Society of Japan*, 90(3):032001, 2021.

[54] Dominic W. Berry, Andrew M. Childs, Richard Cleve, Robin Kothari, and Rolando D. Somma. Exponential improvement in precision for simulating sparse Hamiltonians. *Proc. Annu. ACM Symp. Theory Comput.*, pages 283–292, 2014.

[55] Dominic W Berry. High-order quantum algorithm for solving linear differential equations. *Journal of Physics A: Mathematical and Theoretical*, 47(10):105301, feb 2014.

[56] Michael Lubasch, Jaewoo Joo, Pierre Moinier, Martin Kiffner, and Dieter Jaksch. Variational quantum algorithms for nonlinear problems. *Phys. Rev. A*, 101:010301, Jan 2020.

[57] Emanuel Knill, Gerardo Ortiz, and Rolando D. Somma. Optimal quantum measurements of expectation values of observables. *Phys. Rev. A*, 75:012328, Jan 2007.

[58] Daochen Wang, Oscar Higgott, and Stephen Brierley. Accelerated variational quantum eigensolver. *Phys. Rev. Lett.*, 122:140504, Apr 2019.

[59] Thomas Häner, Martin Roetteler, and Krysta M. Svore. Optimizing Quantum Circuits for Arithmetic. 2018.

[60] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with numpy. *Nature*, 585(7825):357–362, Sep 2020.

[61] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018.

[62] H. Risken. *The Fokker-Planck Equation*. Springer-Verlag, Berlin, Heidelberg, 1996.

[63] Ana Martin, Bruno Candelas, Ángel Rodríguez-Rozas, José D. Martín-Guerrero, Xi Chen, Lucas Lamata, Román Orús, Enrique Solano, and Mikel Sanz. Toward pricing financial derivatives with an ibm quantum computer. *Phys. Rev. Research*, 3:013167, Feb 2021.

[64] 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.

[65] Sergi Ramos-Calderer, Adrián Pérez-Salinas, Diego García-Martín, Carlos Bravo-Prieto, Jorge Cortada, Jordi Planagumà, and José I. Latorre. Quantum unary approach to option pricing. *Phys. Rev. A*, 103:032414, Mar 2021.

[66] Almudena Carrera Vazquez and Stefan Woerner. Efficient State Preparation for Quantum Amplitude Estimation. may 2020.

[67] Hao Tang, Anurag Pal, Tian-Yu Wang, Lu-Feng Qiao, Jun Gao, and Xian-Min Jin. Quantum computation for pricing the collateralized debt obligations. *Quantum Engineering*, 3(4):e84, 2021.

[68] Shouvanik Chakrabarti, Rajiv Krishnakumar, Guglielmo Mazzola, Nikitas Stamatopoulos, Stefan Woerner, and William J. Zeng. A Threshold for Quantum Advantage in Derivative Pricing. *Quantum*, 5:463, June 2021.

[69] Dong An, Noah Linden, Jin-Peng Liu, Ashley Montanaro, Changpeng Shao, and Jiasu Wang. Quantum-accelerated multilevel Monte Carlo methods for stochastic differential equations in mathematical finance. *Quantum*, 5:481, June 2021.

[70] Kenji Kubo, Yuya O. Nakagawa, Suguru Endo, and Shota Nagayama. Variational quantum simulations of stochastic differential equations. *Physical Review A*, 103(5):052425, 2021.

[71] Stefan Woerner and Daniel J. Egger. Quantum risk analysis. *npj Quantum Information*, 5(1):15, 2019.

[72] Daniel J. Egger, Ricardo García Gutiérrez, Jordi Cahué Mestre, and Stefan Woerner. Credit risk analysis using quantum computers. *IEEE Transactions on Computers*, 70(12):2136–2145, 2021.

[73] Koichi Miyamoto. Quantum algorithm for calculating risk contributions in a credit portfolio, 2022.

[74] Patrick Rebentrost and Seth Lloyd. Quantum computational finance: quantum algorithm for portfolio optimization, 2018.

[75] Iordanis Kerenidis, Anupam Prakash, and Dániel Szilágyi. Quantum algorithms for portfolio optimization. In *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*, AFT '19, page 147–155, New York, NY, USA, 2019. Association for Computing Machinery.

[76] Mark Hodson, Brendan Ruck, Hugh Ong, David Garvin, and Stefan Dulman. Portfolio rebalancing experiments using the quantum alternating operator ansatz, 2019.

[77] Román Orús, Samuel Mugel, and Enrique Lizaso. Quantum computing for finance: Overview and prospects. *Reviews in Physics*, 4:100028, 2019.

[78] Daniel J. Egger, Claudio Gambella, Jakub Marecek, Scott McFaddin, Martin Mevissen, Rudy Raymond, Andrea Simonetto, Stefan Woerner, and Elena Yndurain. Quantum computing for finance: State-of-the-art and future prospects. *IEEE Transactions on Quantum Engineering*, 1:1–24, 2020.

[79] Adam Bouland, Wim van Dam, Hamed Joorati, Iordanis Kerenidis, and Anupam Prakash. Prospects and challenges of quantum finance, 2020.

[80] Curt Randall Domingo Tavella. *Pricing Financial Instruments: The Finite Difference Method*. Wiley, 2000.

[81] Daniel J. Duffy. *Finite Difference Methods in Financial Engineering: A Partial Differential Equation Approach*. Wiley, 2006.

[82] Tao Xin, Shijie Wei, Jianlian Cui, Junxiang Xiao, Iñigo Arrazola, Lucas Lamata, Xiangyu Kong, Dawei Lu, Enrique Solano, and Guilu Long. Quantum algorithm for solving linear differential equations: Theory and experiment. *Phys. Rev. A*, 101:032307, Mar 2020.

[83] Andrew M. Childs and Jin-Peng Liu. Quantum spectral methods for differential equations. *Communications in Mathematical Physics*, 375(2):1427–1457, Apr 2020.

[84] Ashley Montanaro and Sam Pallister. Quantum algorithms and the finite element method. *Phys. Rev. A*, 93:032324, Mar 2016.

[85] F. Fillion-Gourdeau and E. Lorin. Simple digital quantum algorithm for symmetric first-order linear hyperbolic systems. *Numerical Algorithms*, 82(3):1009–1045, 2019.

[86] Pedro C. S. Costa, Stephen Jordan, and Aaron Ostrander. Quantum algorithm for simulating the wave equation. *Phys. Rev. A*, 99:012323, Jan 2019.

[87] Shengbin Wang, Zhimin Wang, Wendong Li, Lixin Fan, Zhiqiang Wei, and Yongjian Gu. Quantum fast poisson solver: the algorithm and complete and modular circuit design. *Quantum Information Processing*, 19(6):170, 2020.

[88] Andrew M. Childs, Jin-Peng Liu, and Aaron Ostrander. High-precision quantum algorithms for partial differential equations. *Quantum*, 5:574, November 2021.

[89] Noah Linden, Ashley Montanaro, and Changpeng Shao. Quantum vs. classical algorithms for solving the heat equation. 2020.

[90] Yohichi Suzuki, Shumpei Uno, Rudy Raymond, Tomoki Tanaka, Tamiya Onodera, and Naoki Yamamoto. Amplitude estimation without phase estimation. *Quantum Inf. Process.*, 19(2):1–17, 2020.

[91] Scott Aaronson and Patrick Rall. *Quantum Approximate Counting, Simplified*, pages 24–32. 2020.

[92] Dmitry Grinko, Julien Gacon, Christa Zoufal, and Stefan Woerner. Iterative quantum amplitude estimation. *npj Quantum Information*, 7(1):52, Mar 2021.

[93] Kouhei Nakaji. Faster amplitude estimation. 20(13-14):1109–1123, 2020.

[94] Ville Bergholm, Juha J. Vartiainen, Mikko Möttönen, and Martti M. Salomaa. Quantum circuits with uniformly controlled one-qubit gates. *Phys. Rev. A*, 71:052330, May 2005.

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

[96] Martin Plesch and Časlav Brukner. Quantum-state preparation with universal gate decompositions. *Phys. Rev. A - At. Mol. Opt. Phys.*, 83(3):1–5, 2011.

[97] Raban Iten, Roger Colbeck, Ivan Kukuljan, Jonathan Home, and Matthias Christandl. Quantum circuits for isometries. *Physical Review A*, 93(3):032318, 2016.

[98] Daniel K Park, Francesco Petruccione, and June-Koo Kevin Rhee. Circuit-based quantum random access memory for classical data. *Scientific reports*, 9(1):1–8, 2019.

[99] Israel F Araujo, Daniel K Park, Francesco Petruccione, and Adenilton J da Silva. A divide-and-conquer algorithm for quantum state preparation. *Scientific reports*, 11(1):1–12, 2021.

[100] Peter A. Forsyth David M. Pooley, Kenneth R.Vetzal. Convergence remedies for non-smooth payoffs in option pricing. *Journal of Computational Finance*, 6:25–40, 2003.

[101] Nat Leung Christina Christara. Analysis of quantization error in financial pricing via finite difference methods. *SIAM J. Numerical Analysis*, 56:1731–1757.

[102] Ryan Babbush, Jarrod R. McClean, Michael Newman, Craig Gidney, Sergio Boixo, and Hartmut Neven. Focus beyond quadratic speedups

for error-corrected quantum advantage. *PRX Quantum*, 2:010103, Mar 2021.

[103] Koichi Miyamoto and Kenji Shiohara. Reduction of qubits in a quantum algorithm for monte carlo simulation by a pseudo-random-number generator. *Phys. Rev. A*, 102:022424, Aug 2020.

[104] Kazuya Kaneko, Koichi Miyamoto, Naoyuki Takeda, and Kazuyoshi Yoshino. Quantum speedup of Monte Carlo integration with respect to the number of dimensions and its application to finance. *Quantum Information Processing*, 20(5):185, 2021.

[105] Almudena Carrera Vazquez and Stefan Woerner. Efficient state preparation for quantum amplitude estimation. *Phys. Rev. Applied*, 15:034027, Mar 2021.

[106] Koichi Miyamoto. Bermudan option pricing by quantum amplitude estimation and Chebyshev interpolation. *EPJ Quantum Technology*, 9(1):3, 2022.

[107] Dylan Herman, Cody Googin, Xiaoyuan Liu, Alexey Galda, Ilya Safro, Yue Sun, Marco Pistoia, and Yuri Alexeev. A survey of quantum computing for finance, 2022.

[108] Koichi Miyamoto and Kenji Kubo. Pricing multi-asset derivatives by finite-difference method on a quantum computer. *IEEE Transactions on Quantum Engineering*, 3:1–25, 2022.

[109] Juan Carlos Garcia-Escartin and Pedro Chamorro-Posada. swap test and hong-ou-mandel effect are equivalent. *Phys. Rev. A*, 87:052330, May 2013.

[110] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge University Press, USA, 3 edition, 2007.

[111] Naoto Kunitomo and Masayuki Ikeda. Pricing options with curved boundaries1. *Mathematical Finance*, 2(4):275–298, 1992.

[112] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa,

Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[113] Gustaf Söderlind. The logarithmic norm. history and modern theory. *BIT Numerical Mathematics*, 46(3):631–652, Sep 2006.

[114] L. Beghin and E. Orsingher. On the maximum of the generalized brownian bridge. *Lithuanian Mathematical Journal*, 39(2):157–167, Apr 1999.

[115] R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, 1990.

[116] Vlatko Vedral, Adriano Barenco, and Artur Ekert. Quantum networks for elementary arithmetic operations. *Phys. Rev. A*, 54:147–153, Jul 1996.

[117] Thomas G. Draper. Addition on a quantum computer, 2000.

[118] Steven A. Cuccaro, Thomas G. Draper, Samuel A. Kutin, and David Petrie Moulton. A new quantum ripple-carry addition circuit. *arXiv e-prints*, pages quant–ph/0410184, October 2004.

[119] Yasuhiro Takahashi and Noboru Kunihiro. A linear-size quantum circuit for addition with no ancillary qubits. *Quantum Info. Comput.*, 5(6):440–448, sep 2005.

[120] 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, jul 2006.

[121] Yasuhiro Takahashi, Seiichiro Tani, and Noboru Kunihiro. Quantum addition circuits and unbounded fan-out. *Quantum Info. Comput.*, 10(9):872–890, sep 2010.

[122] J J Álvarez-Sánchez, J V Álvarez-Bravo, and L M Nieto. A quantum architecture for multiplying signed integers. *Journal of Physics: Conference Series*, 128:012013, aug 2008.

[123] Yasuhiro Takahashi and Noboru Kunihiro. A fast quantum circuit for addition with few qubits. *Quantum Info. Comput.*, 8(6):636–649, jul 2008.

[124] Himanshu Thapliyal. *Mapping of Subtractor and Adder-Subtractor Circuits on Reversible Quantum Gates*, pages 10–34. Springer Berlin Heidelberg, Berlin, Heidelberg, 2016.

[125] Himanshu Thapliyal and Nagarajan Ranganathan. Design of efficient reversible logic-based binary and bcd adder circuits. *J. Emerg. Technol. Comput. Syst.*, 9(3), oct 2013.

[126] H. V. Jayashree, Himanshu Thapliyal, Hamid R. Arabnia, and V. K. Agrawal. Ancilla-input and garbage-output optimized design of a reversible quantum integer multiplier. *The Journal of Supercomputing*, 72(4):1477–1493, 2016.

[127] Edgard Muñoz-Coreas and Himanshu Thapliyal. Quantum circuit design of a t-count optimized integer multiplier. *IEEE Transactions on Computers*, 68(5):729–739, 2019.

[128] Alireza Khosropour, Hossein Aghababa, and Behjat Forouzandeh. Quantum division circuit based on restoring division algorithm. In *2011 Eighth International Conference on Information Technology: New Generations*, pages 1037–1040, 2011.

[129] Sayanton Vhaduri Dibbo, Hafiz Md. Hasan Babu, and Lafifa Jamal. An efficient design technique of a quantum divider circuit. In *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 2102–2105, 2016.

[130] Himanshu Thapliyal, T. S. S. Varun, Edgard Munoz-Coreas, Keith A. Britt, and Travis S. Humble. Quantum circuit designs of integer division optimizing t-count and t-depth. In *2017 IEEE International Symposium on Nanoelectronic and Information Systems (iNIS)*, pages 123–128, 2017.

[131] Edgard Muñoz Coreas and Himanshu Thapliyal. T-count and qubit optimized quantum circuit design of the non-restoring square root algorithm. *J. Emerg. Technol. Comput. Syst.*, 14(3), oct 2018.

[132] XIAOYU LI, GUOWU YANG, CARLOS MANUEL TORRES, DESHENG ZHENG, and KANG L. WANG. A class of efficient quantum incrementer gates for quantum circuit synthesis. *International Journal of Modern Physics B*, 28(01):1350191, 2014.

Acknowledgements

First, I would like to thank Prof. Fujii for supporting my doctoral program in spite of his busy schedule. Through our discussions, I found insights and ideas in quantum algorithms. I believe this will be valuable in the future.

Second, I would also like to thank Prof. Miyamoto for co-authoring two papers. Without his collaboration, I would not be able to obtain a deep understanding of quantum algorithms for financial engineering.

The discussions with Prof. Mitarai have deepened my understanding of the quantum machine learning. He also commented on our paper many times, and thanks to them, our paper was brushed up.

I would also like to thank the members of Mercari, Inc. In particular, Dr. Nagayama gave me useful advice on my research and how to progress in my doctoral program. I also benefited from discussions with Mr. Teramoto in the daily research work.

I appreciate Dr. Nakagawa of QunaSys, Inc. and Dr. Endo of NTT Laboratories, Inc. Their support helped me obtain a deep understanding of variational quantum computation and allowed us to write the paper.

I would also like to thank Prof. Kitagawa and Prof. Yamamoto for reviewing my dissertation.

I appreciate all the members of Fujii Lab. Especially with Mr. Shirai and Mr. Kawase, I was able to conduct interesting research through discussions.

Finally, I thank my dear family for their support.