NExt ApplicationS of Quantum Computing

D5.1: Review of state-of-the-art for Pricing and Computation of VaR

Document Properties

Contract Number	951821
Contractual Deadline	M9 (31/05/2021)
Dissemination Level	Public
Nature	Report
Editors	María Nogueiras, HSBC
Authors	María Nogueiras, HSBC Gustavo Ordóñez Sanz, HSBC Carlos Vázquez Cendón, UDC Álvaro Leitao Rodríguez, UDC Alberto Manzano Herrero, UDC Daniele Musso, CESGA Andrés Gómez, CESGA
Reviewers	Vedran Dunjko, Leiden University (ULEI) Antonio Villalpando, Irish Center for Hi-End Computing (ICHEC)
Date	24/05/2021
Keywords	(Quantum) Finance, Quantum Amplitude Estimation, QCoin, Derivatives Pricing, Risk Measures, VaR, CVaR
Status	Final
Release	2.0

This project has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No. 951821

History of Changes

Release	Date	Author, Organisation	Description of Changes
0.1	02/04/2021	María Nogueiras, HSBC Gustavo Ordóñez Sanz, HSBC Carlos Vázquez Cendón, UDC Álvaro Leitao Rodríguez, UDC Alberto Manzano Herrero, UDC Daniele Musso, CESGA Andrés Gómez, CESGA	First draft version
0.2	11/04/2021	María Nogueiras, HSBC Gustavo Ordóñez Sanz, HSBC Carlos Vázquez Cendón, UDC Andrés Gómez, CESGA	Review chapter 3. Rewording of section 4.3.1
0.3	16/04/2021	María Nogueiras, HSBC Gustavo Ordóñez Sanz, HSBC Carlos Vázquez Cendón, UDC Álvaro Leitao Rodríguez, UDC Alberto Manzano Herrero, UDC Daniele Musso, CESGA Andrés Gómez, CESGA	Review of all sections
0.4	22/04/2021	Gustavo Ordóñez Sanz, HSBC	Executive summary, introduction and conclusions
1.0	29/04/2021	María Nogueiras, HSBC Gustavo Ordóñez Sanz, HSBC Carlos Vázquez Cendón, UDC Álvaro Leitao Rodríguez, UDC Alberto Manzano Herrero, UDC Daniele Musso, CESGA Andrés Gómez, CESGA	Final review of all sections
2.0	24/05/2021	María Nogueiras, HSBC Gustavo Ordóñez Sanz, HSBC Carlos Vázquez Cendón, UDC Álvaro Leitao Rodríguez, UDC Alberto Manzano Herrero, UDC Daniele Musso, CESGA Andrés Gómez, CESGA	Revised final review after internal reviewers feedback

Table of Contents

1. Executive Summary

Quantum Computing commenced in 1980's with the pioneering work of Paul Benioff (Benioff, 1980) who proposed a quantum mechanical Turing machine. These ideas were also explored by the likes of Richard Feynman (Feynman, 1982) and Yuri Manin (Manin, 1980) who suggested that quantum computers could provide advantage over classical computers in certain tasks, such as the simulation of physical systems. In 1994, Peter Shor (Shor, 1994) published a groundbreaking paper demonstrating that a quantum algorithm could be used to for large integer factorisation in polynomial time, that is, exponentially faster than the best known classical algorithms. This was followed by Lov K. Grover who proposed a quantum computing algorithm that promised a quadratic speed up over database searches (L. K. Grover, 1996).

Advances in Quantum Computing hardware technology in recent years have been accompanied by the acceleration on the development of quantum computing algorithms with applications across many different use cases in different industry sectors: Automotive, Energy, Logistics, Pharma, Chemical/Manufacturing and the Financial Services Industry. One of the use cases in Finance comes from the application of Quantum Computing for Derivative Pricing and Derivative Risk Management. The purpose of this document is to provide a summary of the "state of the art" for these applications.

However, it is important to note that the currently available quantum computers have limited number of qubits and these suffer from high levels of "noise" which limits the depth and length of the quantum circuits that can be implemented in real hardware. Therefore, near-term applications focus on implementations of quantum algorithms in Noisy Intermediate-Scale Quantum (NISQ) computers (Preskill, 2018).

Derivatives contract form one of the fundamental pillars of modern financial markets and are routinely traded by both financial institutions and traders with a variety of objectives, such as financial risk hedging. A simple example of financial derivative is a European stock option. This contract provides the derivative holder with the right to purchase or sell the stock at some time in the future for a fixed price agreed today. Hence, providing with potential upside (should the stock increase in price at maturity) while limiting the investor's downside.

Derivatives pricing theory is the branch of financial mathematics that covers the fair valuation of financial derivatives such as options. This framework assumes that the underlying security (e.g. a single stock) follows some random (stochastic) process. The price of the derivative hence depends on the particular realisation of such process at a given point in time (e.g. the option maturity). The best known example of option pricing model is the Black-Scholes model (Black, 1976). This model proves that a fair value of an option can be derived under certain assumptions (e.g. absence of arbitrage, continuous and unlimited long and short trading).

However, in general the Black-Sholes model is too simplistic to fit actual quoted prices in the market and other more complex models are used instead, at the cost of requiring numerical approximations to find the fair price of the derivative. Two main numerical approaches are currently used in the industry, Monte Carlo-based simulation techniques and partial differential equation (PDEs) approaches. The key advantage of the former is that it is easy to implement, very general and scales well with the dimension of the problem. On the other hand, Monte Carlo simulation tends to converge slowly to the required solution. It is also difficult to obtain risk sensitivities (i.e. how the derivative price depends on changes to the price underlying) using Monte Carlo. PDEs approaches are generally faster and permit the easy calculation of risk sensitives, however it is usually a difficult problem to solve PDEs for more complex derivatives, specially those that depend on several risk factors (curse of dimensionality). Monte Carlo simulation is therefore the tool of choice for financial risk management where risk metrics need to be estimated at the portfolio level where thousands of derivatives need to be covered.

Quantum computing, and in particular the Quantum Amplitude Estimation (QAE) algorithm promises a potential quadratic speed up over classical Monte Carlo approaches but maintaining its main advantages: easy of implementation and linear scaling to higher dimensions. This document covers these topics in more detail and presents some new results, such as the application of the "Quantum Coin" algorithms as an alternative to QAE.

Despite the highly promising advantages of quantum computing for derivative pricing and risk management, huge challenges remain open for real-world applications. Some of them are technological, such as the relative small number of qubits currently available and the fact that these are "noisy" (i.e. not always reliable).

Others are more theoretical, such as the lack of understanding on how to load classical information (such as probability distributions) to quantum registers, or how to represent relatively complex pay-off functions with quantum circuits that are as small as possible.

2. Introduction

NExt ApplicationS of Quantum Computing (NEASQC) project has the main objective of developing industrial use cases that can work on current Near Intermediate Scale Quantum (NISQ) machines and that can create a rich set of quantum algorithms and libraries. To achieve this aim, the project proposed 9 uses cases which are grouped in three different topics: chemistry, machine learning and optimization, and symbolic AI and graph algorithmics. One of the industrial use cases is Quantum Financial Applications Use Case (UC5). UC5 aims to develop efficient algorithms that could either substitute or redefine Monte Carlo (MC) techniques in NISQ computers.

Quantum computing for financial applications is a hot topic, with multiple and varied possibilities (see (Orús et al., 2019) and (Bouland et al., 2020) for recent reviews). The reason for exploring this field of quantum computing is because current algorithms used by financial institutions demand a high computing capacity because many of the models do not have analytical solution, needing High Performance Computing to achieve a fast response of the numerical approximations. Quantum computing promises for some cases faster responses that can provide solutions to up-to-now intractable problems. This fact motivates that quantitative finance explores its usage.

The first set of algorithms to be explored inside UC5 corresponds to pricing of financial derivatives and Value-at-Risk (VaR) computation. In the first case, the objective is to provide the price of an option that gives the right to sell or buy an asset in a future date at a fixed price. In the second case, the VaR is a risk measure so that the objective is to minimize the losses of future operations. This report reviews the main concepts and classical algorithms to address both mathematical problems, as well as the recent advances to tackle these problems with quantum computing.

The document is divided into two main parts. The first “Classical Methods” Chapter 3 provides a quick overview of the approaches currently used to price and manage the risk of financial derivatives. Financial derivatives are contracts whose pay off depends on a particular underlying asset, say a particular company stock, a foreign exchange, or a given currency yield curve. A simple example of a derivative contract is a European Call Option on a single equity stock. This contract gives the holder the right to purchase the stock at some point in the future for a fixed price agreed today. This means that, if at maturity the stock price increases, the option holder can make a profit by purchasing the stock at the previously agreed price and selling it in the market at the current, higher, price.

The chapter starts with a review of some basic derivative contracts and their pay off functions. It continues with some modelling approaches and finally covers two general numerical approaches for derivative pricing: Monte Carlo and Partial Differential Equations- PDEs (including machine learning approaches). In the second part of the chapter financial risk management concepts are covered such as Market Risk and Credit Portfolio Management, as well as key risk portfolio metrics such as Value-at-Risk and Conditional Value-at-Risk (or Shortfall). Factor models, a key dimension reduction techniques are also covered. The chapter ends with a review of some of the most important numerical methods in financial risk management.

The second part of the document “Quantum toolkit for Finance” Chapter 4 is a review of the currently known approaches to implement derivative pricing and derivative risk management in quantum computers. Special focus is paid to quantum alternatives to classical Monte Carlo. Despite its relative simplicity, Monte Carlo approaches are widely used for derivative pricing and derivative risk management in financial institutions mainly due to their general purpose and larger resilience to the “curse of dimensonality”.

One of the key topics covered is the Quantum Amplitude Estimation (QAE) algorithm which is a well known quantum alternative to Monte Carlo approaches to derivative pricing and which promises a quadratic speed up over its classical counterpart. In particular, recent enhancements of this algorithm that make it more suitable for NISQ quantum technology are covered.

A novel approach proposed by the authors and which represents an alternative to QAE is presented in Section 4.1.3. This approach maintains the promise of quadratic speed up although with a simpler implementation at the hardware level.

Later in the chapter, quantum methods for solving the derivative pricing model using PDEs are also discussed.

The chapter concludes with a review of the key remaining challenges for these types of quantum algorithms

to deliver on their potential advantage. Namely, the “loading problem”, that is the efficient loading of general probability distributions and pay-off functions to qubit registers. Finally, the “scaling to real-world problem size” section briefly covers the remaining challenges on how to implement these types of approaches for real-world applications.

3. Classical methods

3.1. Pricing of financial derivatives

3.1.1. Financial derivatives and options

In financial markets, a *derivative* (with price at time t: v_t) is a financial contract with future cash flows depending on the performance of one of several *assets* (usually referred as *underlying asset*, or simply *underlying*, S_t).

Examples of underlying assets are stocks, interest rates, exchange rates, credit spreads ... Depending on the main underlying *risk factor*, we have equity derivatives, interest rate derivatives, FX derivatives, credit derivatives, etc.

Options (or contingent claims, or non linear derivatives) are derivatives whose payoffs are non linear functions of the underlying.

The cash flow of the option, as a function of the underlying asset, is called *payoff*, and is received by the option holder at the expiring date of the option contract, or before under certain circumstances. The payoff is usually defined by a mathematical expression, $v_T = h(T, S_T)$, where h is a function that depends on the expiry date T and the price of the underlying at the expiry date S_T. At any time $0 < t < T$, the option price $v_t = V(t, S_t)$, where V is a function depending on time t and the price of the underlying S_t that represents the premium the buyer has to pay to get the rights associated to the option contract.

In *derivative pricing* mathematical models define the relationship between underlying asset and option prices, with the objective of calculating the fair value of the options. Typically, standard (more liquid) options are used for calibration of the model parameters.

While pricing is related to *expected* cashflows, risk computation is related to a *tail* scenarios.

In the following, some of the most popular families of options based on their payoffs are described. Similarly, some of the mathematical models are defined later with the most common numerical techniques to solve them.

3.1.1.1. European options

A European vanilla option on a underlying asset gives the right but not the obligation to buy or sell the asset at a given date in the future (T or expiry date) and for a fixed price (which is commonly known as the strike price K) (Hull, 1997). When the option confers the right to buy at strike price K at time T it is referred to as a *call option*, and when it gives the right to sell it is called a *put option*.

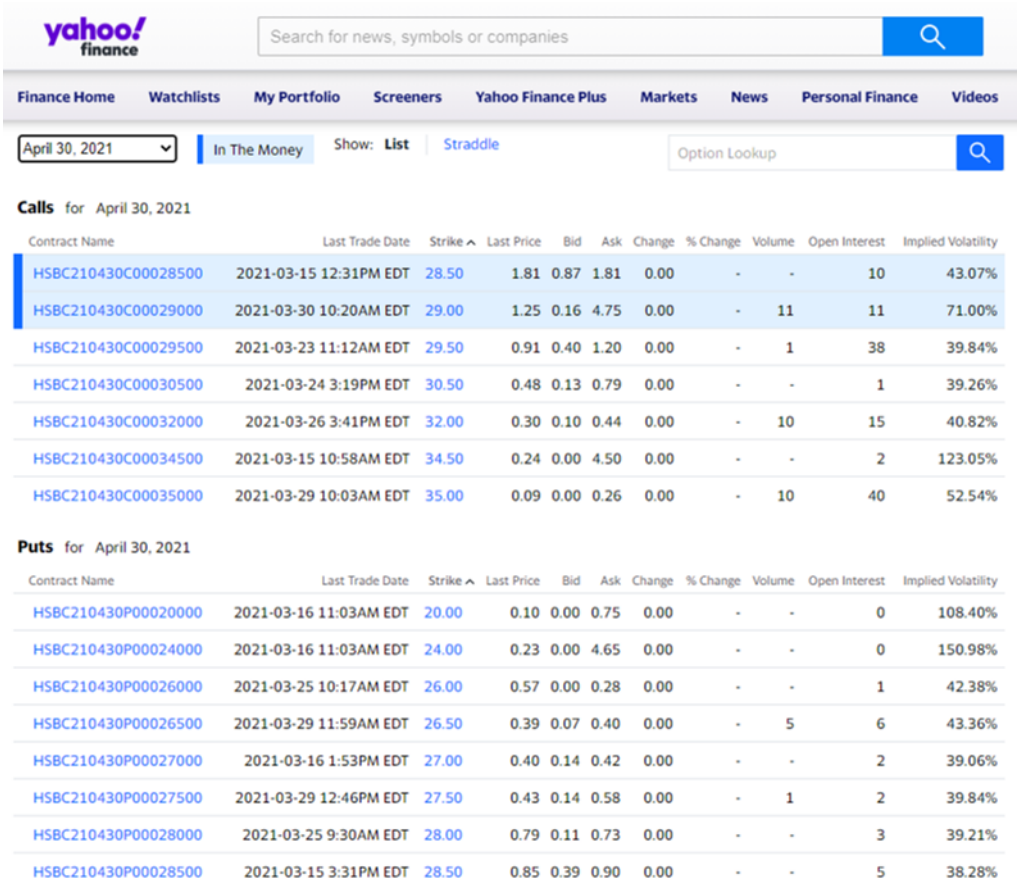
Denoting S_t as the underlying asset value at time t with $t \in [0, T]$, the payoffs of the call and put options at expiry (or maturity) date T can be calculated as:

$$c_T = \max(S_T - K, 0), \quad p_T = \max(K - S_T, 0) \tag{3.1}$$

Notice that at maturity date $t = T$ these values are known, since the payoff is deterministic and S_T is known. The option pricing problem is to calculate the fair value of the contracts at $t = 0$, c_0 and p_0, that will be a function of the known underlying value S_0 and other model parameters (risk-free rate, time to maturity,...).

For example, in Figure 1 options on HSBC stock with expiry date April 30^{th}, 2021 are shown[1]. The value of the HSBC stock at that date was 29.16, The calls marked in blue were *in the money* on that day since the strike price is below the current value of the stock, however should the value of the stock fall below the strike price at maturity these options would expire *out the money*, i.e. with a value of zero. When the value of the underlying is exactly that of the strike, the option is known to be *at the money*. In Figure 1, the *last price* column refers to the most recent value of the option in the market.

[1] Data from Yahoo Financials, consulted on Apr. 2^{nd}, 2021.



yahoo! finance

Search for news, symbols or companies

Finance Home | Watchlists | My Portfolio | Screeners | Yahoo Finance Plus | Markets | News | Personal Finance | Videos

April 30, 2021 | In The Money | Show: List | Straddle | Option Lookup

Calls for April 30, 2021

Contract Name	Last Trade Date	Strike	Last Price	Bid	Ask	Change	% Change	Volume	Open Interest	Implied Volatility
HSBC210430C00028500	2021-03-15 12:31PM EDT	28.50	1.81	0.87	1.81	0.00	-	-	10	43.07%
HSBC210430C00029000	2021-03-30 10:20AM EDT	29.00	1.25	0.16	4.75	0.00	-	11	11	71.00%
HSBC210430C00029500	2021-03-23 11:12AM EDT	29.50	0.91	0.40	1.20	0.00	-	1	38	39.84%
HSBC210430C00030500	2021-03-24 3:19PM EDT	30.50	0.48	0.13	0.79	0.00	-	-	1	39.26%
HSBC210430C00032000	2021-03-26 3:41PM EDT	32.00	0.30	0.10	0.44	0.00	-	10	15	40.82%
HSBC210430C00034500	2021-03-15 10:58AM EDT	34.50	0.24	0.00	4.50	0.00	-	-	2	123.05%
HSBC210430C00035000	2021-03-29 10:03AM EDT	35.00	0.09	0.00	0.26	0.00	-	10	40	52.54%

Puts for April 30, 2021

Contract Name	Last Trade Date	Strike	Last Price	Bid	Ask	Change	% Change	Volume	Open Interest	Implied Volatility
HSBC210430P00020000	2021-03-16 11:03AM EDT	20.00	0.10	0.00	0.75	0.00	-	-	0	108.40%
HSBC210430P00024000	2021-03-16 11:03AM EDT	24.00	0.23	0.00	4.65	0.00	-	-	0	150.98%
HSBC210430P00026000	2021-03-25 10:17AM EDT	26.00	0.57	0.00	0.28	0.00	-	-	1	42.38%
HSBC210430P00026500	2021-03-29 11:59AM EDT	26.50	0.39	0.07	0.40	0.00	-	5	6	43.36%
HSBC210430P00027000	2021-03-16 1:53PM EDT	27.00	0.40	0.14	0.42	0.00	-	-	2	39.06%
HSBC210430P00027500	2021-03-29 12:46PM EDT	27.50	0.43	0.14	0.58	0.00	-	1	2	39.84%
HSBC210430P00028000	2021-03-25 9:30AM EDT	28.00	0.79	0.11	0.73	0.00	-	-	3	39.21%
HSBC210430P00028500	2021-03-15 3:31PM EDT	28.50	0.85	0.39	0.90	0.00	-	-	5	38.28%

***Figure 1**: Options on HSBC stocks (the underlying assets) in Yahoo Financials for expiry date of Apr. 30^{th}, 2021*

3.1.1.2. American options

Unlike European options, where the option can only be exercised at expiry date T and receive the payoff (3.1), American options give the holder the right to exercise the call (put) option, i.e. the right to buy (or sell) the underlying stoke at the strike value at any time $t_e \in [0, T]$ and receive:

$$c_{t_e} = \max\left(S_{t_e} - K, 0\right), \quad p_{t_e} = \max\left(K - S_{t_e}, 0\right), \tag{3.2}$$

which is referred to as the exercise value of the call or the put. Therefore, the value of an American options is always greater than or equal to its exercise value:

$$c_t \geq \max\left(S_t - K, 0\right), \quad p_t \geq \max\left(K - S_t, 0\right), \tag{3.3}$$

otherwise there would be arbitrage opportunities, since a trader could make a riskless profit by selling (or buying) American and European calls/puts on the stock. While arbitrage opportunities do exist in financial markets, these are normally short lived and not usually material, therefore option pricing theory assumes the absence of arbitrage opportunities.

3.1.1.3. Basket options

In previous sections we introduced examples of options that depend on just one underlying asset. Options that depend on a set of assets are also traded in the financial markets and are known as *basket* options. The payoff of a vanilla basket option depends on the value of a set of assets at maturity, that is:

$$c_T = f^c(T, S_T^1, \ldots, S_T^d), \quad p_T = f^p(T, S_T^1, \ldots, S_T^d). \tag{3.4}$$

Some examples of specific payoffs of basket options are:

$$\begin{aligned}
\text{Call spread option:} \quad & F(S^1, S^2) = \max(S^1 - S^2 - K, 0) \\
\text{Put spread option:} \quad & F(S^1, S^2) = \max(K - (S^1 - S^2), 0) \\
\text{"Best of" type option:} \quad & F(S_T^1, \ldots, S_T^d) = g(\max(S^1, \ldots, S_T^d)) \\
\text{"Worst of" type option:} \quad & F(S^1, \ldots, S_T^d) = g(\min(S^1, \ldots, S_T^d))
\end{aligned}$$

Note that basket options can be either American or European style options, i.e. they can include early exercise opportunity or not.

At this introductory level, we are going to focus in vanilla contracts only. For vanilla options, the payoffs only depend on the value of the underlying at expiry date T. Examples of more complicated options are Asian, where the payoff is determined by the average underlying value or barrier options, where the payoff depends on if the underlying option value has or not has passed some limits. These non-vanilla options will not being considered here.

3.1.2. Some models for the value of underlying assets

One of the key ingredients in option pricing is the choice of the dynamics of the underlying stochastic factors. Although there are a lot of possible choices of factors and dynamics that could be taken into account, in this review we will focus on the classical Black-Scholes (Black & Scholes, 1973) and Heston dynamics (Heston, 1993) for the underlying asset evolution. This framework can easily be extended to the values of a set of assets when dealing with basket options.

Notice that these models and stochastic evolution are popular in the context of the *Equity* asset class.

3.1.2.1. Black-Scholes model

Black-Scholes model allows to compute the value of a derivative product $v_t = V(t, S_t)$ at time t in terms of the value of the underlying asset S_t at time t (Vázquez, 2010). In the Black-Scholes model one assumes that the dynamics of the underlying value S_t follows a geometric Brownian motion, so that it satisfies the following Stochastic Differential Equation (SDE):

$$dS_t = \mu S_t dt + \sigma S_t dW_t, \tag{3.5}$$

where μ is the drift parameter, σ is the stock volatility and dW_t represents the increment of a Wiener process[2], also called a Brownian motion. This Brownian motion is introduced within the frame of a filtered probability space involving a probability measure P.

In the case of basked options, the Black-Scholes dynamics of each underlying asset satisfies:

$$dS_t^i = \mu^i S_t^i dt + \sigma^i S_t^i dW_t^i, \tag{3.6}$$

where μ^i, σ^i and W_t^i are the corresponding drift, volatility and driving Brownian motion of asset i, for $i = 1, \ldots, d$. Correlations between assets are captured through correlations between their corresponding Brownian motions, that is $dW_t^i dW_t^j = \rho_{ij} dt$, ρ_{ij} being the instantaneous correlation coefficient between asset S_t^i and S_t^j. Note that correlation coefficient can be time dependent.

3.1.2.2. Heston model

Due to some drawbacks in the classical Black-Scholes model to match prices quoted in the markets (or *calibration*), stochastic volatility models replace the constant volatility σ assumed in the BS-model. Considering two separate processes, one for the value S_t and another one for the instantaneous variance ν_t, this provides us with richer dynamics that can fit market observed prices better. One of the most popular

[2] A Wiener process is any real valued continuous-time stochastic process

stochastic volatility models is the Heston model, that assumes the following dynamics for the underlying asset and its variance:

$$\begin{aligned} dS_t &= \mu S_t dt + \sqrt{\nu_t} S_t dW_t^S, \\ d\nu_t &= \kappa(\theta - \nu_t)dt + \chi\sqrt{\nu_t} dW_t^\nu. \end{aligned} \tag{3.7}$$

In the fist equation, μ is the drift parameter and $\sqrt{\nu_t}$ is the stochastic volatility. In the second equation, κ is the mean reversion speed, θ is the long term variance also called the level and χ is referred to as the volatility of the volatility (vol-of-vol). Finally, two correlated Brownian motions W_t^S and W_t^ν are considered with correlation coefficient ρ that capture the dependency between asset value and variance, i.e. $dW_t^S dW_t^\nu = \rho dt$.

Extension to a multi-assets dynamics with d assets can be addressed by considering $2d$ processes corresponding to Heston dynamics for each asset.

3.1.3. Pricing methods

In order to compute the value of European options with one underlying asset, we can use Girsanov theorem[3] to work in the aslo known as risk neutral probability measure Q so that the dynamics of the underlying asset under this measure is obtained by replacing the drift μ by the risk free rate r in 3.5. Under measure Q, the value of the option is a martingale process, that is, the conditional expected value of the discounted value of the derivative is constant through time. Next, we use Ito lemma[4] and the martingale property, so that the option value $v_t = V(t, S_t)$ can be obtained as a conditional expectation of the form:

$$V(t, S_t) = e^{-r(T-t)} \mathbb{E}_Q[h(T, S_T)|\mathcal{F}_t], \tag{3.8}$$

such that S_t is an stochastic process satisfying any of the previous models. Note that r is the discounting rate, h is the payoff function and the term $\mathcal{F}_t$, the filtration, represents the market information (which is assumed to be known) until time t. For further details on probability theory and stochastic calculus we refer to (Mikosch, 1998).

Regardless the pricing problem formulation (conditional expectation, PDEs or other), and the techniques of derivation (Girsanov theorem, dynamic hedging and non arbitrage, etc) risk neutral measure is used for pricing of derivatives.

3.1.3.1. Monte Carlo

The expectation in Equation (3.8) can be written in integral form a *risk-neutral valuation formula*,

$$V(t, S) = e^{-r(T-t)} \int_{\mathbb{R}} h(T, y) f(y|\mathcal{F}_t) dy, \tag{3.9}$$

with $f(\cdot)$ being the *Probability Density Function* (PDF) of the asset value process. Many numerical techniques are based on the solution provided by this risk-neutral valuation formula, in particular by taking advantage of its integral formulation.

As the integral in Equation (3.9) is typically not solvable in analytic form, numerical-based approaches can be developed. Monte Carlo methods are well-known numerical techniques to evaluate integrals. They are based on the analogy between probability and volume. Suppose we need to compute an integral

$$I := \int_C g(x) dx,$$

and we have a technique to draw M independent and identically distributed samples in C, $X_1, X_2, \ldots, X_M$.

[3] In probability theory, the Girsanov theorem describes how the dynamics of stochastic processes is modified when the original measure is changed to an equivalent probability measure. Source: https:en.wikipedia.org/wiki/Girsanov_theorem

[4] A formula to calculate the differential of a time-dependent function of a stochastic process. It is the equivalent in stochastic calculus to the chain rule in the usual calculus.

We then define a Monte Carlo estimator (see (Glasserman, 2004), for further details) as

$$\bar{I}_M := \frac{1}{M}\sum_{m=1}^{M} g(X_m).$$

If g is integrable over C then, by the *strong law of the large numbers*,

$$\bar{I}_M \to I \quad \text{as} \quad M \to \infty,$$

with probability one.

Furthermore, if g is square integrable, we can define the standard deviation of g as

$$s_g := \sqrt{\int_C (g(x) - I)^2 \, dx}.$$

By the *central limit theorem*, it is known that the error of the Monte Carlo estimate, i.e. $I - \bar{I}_M$, is normally distributed with mean 0 and standard deviation $s_g/\sqrt{M}$, where, again, M is the number of sample paths. Therefore, the rate of convergence of the Monte Carlo method is $\mathcal{O}(1/\sqrt{M})$, which is considered slow for many applications.

However, Monte Carlo methods are highly appreciated in computational finance due to their simplicity, flexibility and easy implementation. One of their most important advantages is that these methods are easily extendable to multi-dimensional problems without increasing the rate of convergence. Moreover, the method's simplicity allows for different convergence improvement techniques, such as *variance reduction techniques* (Glasserman, 2004) or *multi-level Monte Carlo* acceleration (Giles, 2015).

When employing Monte Carlo to approximate an expectation, an essential part of the method is the to generation of sample paths. *Random number generators* (RNG) are typically used to generate Monte Carlo paths and they have been studied for many years. Broadly, they can be subdivided into "true", pseudo- and quasi-random generators, and they usually generate uniformly distributed samples. This is key, because when uniform samples between 0 and 1 are available, samples from any distribution can be obtained as long as the *quantile function*, i.e., the inverse of the *Cumulative Distribution Function* (CDF), is known. The procedure is then as follows,

$$F_Z(Z) \overset{\mathrm{d}}{=} U \quad \text{thus} \quad Z_m = F_Z^{-1}(U_m),$$

where F_Z is the CDF, $\overset{\mathrm{d}}{=}$ means equality in the distribution sense, $U \sim \mathcal{U}([0,1])$ and U_m is a sample from $\mathcal{U}([0,1])$. The computational efficiency highly depends on the cost of calculating F_Z^{-1}.

When the "exact" sample generation is not possible, either because the distribution is not available or the inversion is computationally unaffordable, intermediate steps in the numerical scenario simulation can be introduced by means of a time discretization of the associated SDE. Taylor expansion-based discretization schemes are widely employed in quantitative finance. The most representative example is the *Euler-Maruyama method*, the generalization of the Euler method to SDEs is available in the context of the Itô's calculus. Another well-known Itô-Taylor-based simulation scheme is the *Milstein method*, which presents a higher order of convergence than the Euler-Maruyama method. See (Kloeden & Platen, 2013) for further details on numerical methods for SDE.

In order to use Equation (3.8) the expectation of S_t at time T needs to be estimated. To obtain this, it is possible to generate trajectories of S_t from Equation (3.5). These trajectories can be obtained discretizing Equation (3.5) with an Euler-Maruyama scheme, starting from the given value S_0 at $t = 0$:

$$S_{j+1}^m = S_j^m + rS_j^m \Delta t + \sigma S_j^m dW_j, \tag{3.10}$$

$\Delta t = T/(J+1)$ is the uniform time step, $j = 0, \ldots, J+1$ is the index for each time step and $m = 1, ..., M$ is the super index for each trajectory, so that S_j^m approximates the value at time $t_j = j\Delta t$ on the trajectory m.

Once the trajectories have been obtained, it is possible approximate the option value at $t = 0$ in expression

(3.8) by:

$$v_0 = V(0, S_0) \approx \exp(-rT)\,\frac{1}{M}\sum_{m=1}^{M} h(t_{J+1}, S_{J+1}^m). \tag{3.11}$$

It is straight forward to extend Monte Carlo methods to the case of European options on a finite set of correlated assets following the Black-Scholes dynamics (3.6). This can be achieved by computing the trajectories of all involved assets, taking into account correlations by means of correlated Brownian motions that are obtained from independent ones and using, for instance, the Cholesky factorization of the correlation matrix $\rho = (\rho_{ij})$ to generate correlated paths. It is also easy to extend Monte Carlo methods for pricing European options under the Heston model.

The use of Monte Carlo techniques for American options is more complex due to the possibility of early exercise included in these options which makes the problem path dependent. For American options, the formulation in terms of expectations involves the so called optimal stopping time. More precisely, in the case of one underlying asset, the American option value at time t is given by expression:

$$v_t = V(t, S_t) = \max_{\tau\in[t,T]} \mathrm{e}^{-r(T-\tau)}\mathbb{E}_Q[h(\tau, S_\tau)|\mathcal{F}_t], \tag{3.12}$$

where τ denotes a stopping time. A common approach to pricing American options involves a combination of Monte Carlo and regression techniques. Longstaff-Schwartz is a classical example of these methods (Longstaff & Schwartz, 2001).

3.1.3.2. PDEs: Finite differences

The pricing of European options admits an equivalent formulation in terms of Partial Differential Equation (PDE). This formulation can be obtained leveraging the Feynman-Kac theorem that relates expectations of stochastic processes with the solution of PDEs. Alternatively, this formulation can also be derived using dynamic hedging methodologies which involve the use of Ito Lemma, the building of a risk free portfolio and the consideration of absence of arbitrage hypothesis. Under this formulation the function V, such that $v_t = V(t, S_t)$ is the value of the European option, satisfies the following Black-Scholes PDE in $\Omega = (0, T) \times \mathbb{R}^+$:

$$\frac{\partial V}{\partial t} + rS\frac{\partial V}{\partial S} + \frac{\sigma^2 S^2}{2}\frac{\partial^2 V}{\partial S^2} - rV = 0. \tag{3.13}$$

$$V(T, S) = h(T, S) \tag{3.14}$$

The final condition at time T is given by function h and depends on the payoff of the specific product. The solution for this PDE is (Wilmott, 2007)

$$V(t, S) = \frac{e^{-r(T-t)}}{\sigma\sqrt{2\pi(T-t)}}\int_0^\infty \exp\left(-\frac{(\log(S/S') + (r - \frac{\sigma^2}{2})(T-t))^2}{2\sigma^2(T-t)}\right) h(t, S')\frac{dS'}{S'}\,. \tag{3.15}$$

Note that for the case of European vanilla call or put options, the solution of (3.13) has an analytical expression, also known as Black-Scholes formulas:

$$V_c(t, S_t) = S_t\mathcal{N}(0,1)(d_1) - Ke^{-r(T-t)}\mathcal{N}(0,1)(d_2), \tag{3.16}$$

$$V_p(t, S_t) = -S_t\mathcal{N}(0,1)(-d_1) + Ke^{-r(T-t)}\mathcal{N}(0,1)(-d_2), \tag{3.17}$$

with

$$d_1 = \frac{\log(S/K) + (r + \frac{1}{2}\sigma^2)(T-t)}{\sigma\sqrt{T-t}}, \tag{3.18}$$

$$d_2 = \frac{\log(S/K) + (r - \frac{1}{2}\sigma^2)(T-t)}{\sigma\sqrt{T-t}} = d_1 - \sigma\sqrt{T-t}, \tag{3.19}$$

where $\mathcal{N}(0,1)(x)$ is the CDF of the standard normal distribution. For more general payoffs, it is not always possible to find closed form solutions like this one and numerical methods are required.

A common technique to solve PDEs like (3.13) is based on the so called finite differences methods. To use this approach it is necessary to define a bounded domain for the underlying asset so that the initial unbounded domain is truncated. A typical choice for the upper bound is $S_\infty = 4K$, so that $S \in [0, S_\infty]$.

The next step is to define a finite differences mesh by choosing two natural numbers $J > 1$ and $I > 1$, so that the constant time and underlying step sizes are $\Delta t = T/(J+1)$ and $\Delta S = S_\infty/(I+1)$, respectively. Thus, the finite differences mesh nodes are $(t_j, S_i) = (j\Delta t, i\Delta S),\ j = 0, ..., J+1;\ i = 0, ...I+1$. At the mesh nodes, the involved derivatives in the PDE ((3.13)) are approximated as follows:

$$\frac{\partial V}{\partial S}(t_j, S_i) \approx \frac{V(t_j, S_{i+1}) - V(t_j, S_{i-1})}{2\Delta S}, \tag{3.20}$$

$$\frac{\partial^2 V}{\partial S^2}(t_j, S_i) \approx \frac{V(t_j, S_{i+1}) - 2V(t_j, S_i) + V(t_j, S_{i-1})}{(\Delta S)^2}, \tag{3.21}$$

$$\frac{\partial V}{\partial t}(t_j, S_i) \approx \frac{V(t_{j+1}, S_i) - V(t_j, S_i))}{\Delta t}. \tag{3.22}$$

In order to discretize in time, the second order Crank-Nicolson scheme can be used. Once these approximations of the derivatives have been introduced and using the notation $V_{ji} \approx V(t_j, S_i)$, the discretization of Equation (3.13) can be written as the following set of I linear equations for $j = J, J-1, \ldots, 0$:

$$\begin{aligned}
&\frac{V_{j+1,i} - V_{j,i}}{\Delta t} = \\
&\frac{1}{2}\left(-rS_i\frac{V_{j,i+1} - V_{j,i-1}}{\Delta S} - \frac{\sigma^2 S_i^2}{2}\frac{V_{j,i+1} - 2V_{j,i} + V_{j,i-1}}{(\Delta S)^2} + rV_{j,i}\right) \\
&+ \frac{1}{2}\left(-rS_i\frac{V_{j+1,i+1} - V_{j+1,i-1}}{\Delta S} - \frac{\sigma^2 S_i^2}{2}\frac{V_{j+1,i+1} - 2V_{j+1,i} + V_{j+1,i-1}}{(\Delta S)^2} + rV_{j+1,i}\right), \quad i = 1, \ldots, I.
\end{aligned} \tag{3.23}$$

Additionally, boundary conditions depending on the specific payoff are imposed to complete the set of equations. For example, in the case of a call option:

$$V_{j,I+1} = S_\infty - e^{-r(T-t_j)}K, \qquad V_{j,0} = 0. \tag{3.24}$$

Note that for each index j (associated to time t_j) we need to solve a linear system, which can be written in matrix form as $AV_j = b_j$, where V_j is the vector of unknowns containing the option values approximation at time $t = t_j$ for the discrete asset values $S_i,\ i = 0, \ldots, I+1$. Starting from $j = J+1$ the system can be solved sequentially for $j = J, J-1, \ldots, 0$. Thus, the pricing of European options with one underlying asset in a PDEs formulation mainly involves the sequential solution of linear systems.

The extension of the finite differences methodology to include the Heston model or European basket options is straight forward but carry the cost of increasing computational demand with each additional dimension, the so called *curse of dimensionality*. In classical computers, the implementation in multi-CPUs or specific techniques like sparse grids can be used to alleviate this.

In the case of American options, the early exercise opportunity implies the replacement of PDE problem (3.13) by a linear complementarity problem associated to the Black-Scholes differential operator:

$$\mathcal{L}(V) = \frac{\partial V}{\partial t} + rS\frac{\partial V}{\partial S} + \frac{\sigma^2 S^2}{2}\frac{\partial^2 V}{\partial S^2} - rV.$$

More precisely, the complementarity problem is written as (see (Vázquez, 2010) and the references therein):

$$\mathcal{L}(V) \leq 0, \quad V \geq h, \quad \mathcal{L}(V) \cdot (V - h) = 0. \tag{3.25}$$

There is no known analytical solution to this problem and therfore it needs to be solved numerically. When discretizing the complementarity problem using the same approximations as in the European options, we obtain:

$$AV_j \leq b_j, \quad V_j \geq h_j, \quad (AV_j - b_j)^t \cdot (V_j - h_j) = 0, \tag{3.26}$$

where $h_j = h(t_j, .)$ is the vector of exercise values at time t_j in the finite differences asset nodes and super index t denotes the traspose operation. It is easy to see that problem (3.26) can be expressed as a

quadratic optimization with inequality constraints of the form:

$$V_j = \arg\min_{Y \geq h_j} \frac{1}{2} Y^t A Y - b_j^t Y. \tag{3.27}$$

Therefore, numerical methods for solving convex quadratic optimization problems under inequality constraints can be used, such as penalization or duality techniques.

3.1.3.3. PDEs: Artificial Neural Network (ANN) and deep learning

Unsupervised deep learning techniques can be used in derivatives pricing, see (Beck et al., 2021) and the references therein, for applications for solving solving both linear and nonlinear time-dependent PDEs such as the ones presented above.

A general PDE problem can be written as:

$$\begin{aligned} \mathcal{N}_I(v(t,x)) &= 0, \quad x \in \widetilde{\Omega},\, t \in [0,T], \\ \mathcal{N}_B(v(t,x)) &= 0, \quad x \in \partial\widetilde{\Omega},\, t \in [0,T], \\ \mathcal{N}_0(v(t^*,x)) &= 0, \quad x \in \widetilde{\Omega} \text{ and } t^* = 0 \text{ or } t^* = T, \end{aligned} \tag{3.28}$$

where $v(t,x)$ denotes the solution of the PDE, $\mathcal{N}_I(\cdot)$ is a linear or nonlinear time-dependent differential operator, $\mathcal{N}_B(\cdot)$ is a boundary operator, $\mathcal{N}_0(\cdot)$ is an initial or final time operator, $\widetilde{\Omega}$ is a subset of $\mathbb{R}^D$ and $\partial\widetilde{\Omega}$ denotes the boundary on the domain $\widetilde{\Omega}$.

As mentioned before, both European and American option pricing using PDEs can be casted in this formulation and, therefore, solved using unsupervised deep learning. The goal is to obtain $\hat{v}(t,x)$ by minimizing a suitable loss function $L(v)$ over the space of k-times differentiable functions, where k depends on the order of the derivatives in the PDE, i.e.,

$$\arg\min_{v \in \mathcal{C}^k} L(v) = \hat{v}\,, \tag{3.29}$$

where $\hat{v}(t,x)$ denotes the true solution of the PDE.

The solution of the PDEs is approximated with a deep neural network and the accuracy of this approximation can be related to value of the cost function used. The deep neural network consists of an input layer with d neurons, several hidden layers and an ouput layer with a single neuron, representing the entire solution of the PDE. The ANN should approximate the solution, satisfying the restrictions imposed by the PDE and the boundary conditions. A general expression for the cost function, defined in terms of the L^p norm with a given weighting, is defined as follows:

$$\begin{aligned} L(v) = \lambda &\int_\Omega |\, \mathcal{N}_I(v(t,x)) \,|^p \, dxdt \\ &+ (1-\lambda) \int_{\partial\Omega} (|\, \mathcal{N}_B(v(t,x)) \,|^p + |\, \mathcal{N}_0(v(t,x)) \,|^p)\, dxdt, \end{aligned} \tag{3.30}$$

where $\Omega = \widetilde{\Omega} \times [0,T]$, $\partial\Omega$ the boundary of Ω and operators have the form

$$\begin{aligned} &\mathcal{N}_I(v(t,x)) \equiv N(v(t,x)) - F(t,x) \quad \text{in } \Omega\,, \\ &\mathcal{N}_B(v(t,x)) \equiv B(v(t,x)) - G(t,x) \quad \text{on } \partial\widetilde{\Omega}\,, \\ &\mathcal{N}_0(v(t^*,x)) \equiv H(x) - v(t^*,x) \quad \text{in } \widetilde{\Omega} \times t^*, \text{ with } t^* = 0 \text{ or } t^* = T, \end{aligned} \tag{3.31}$$

where N, F, B, G and H are generic functions whose expression depends on the problem at hand.

The parameter $\lambda \in (0,1)$ in the cost functions represents the relative importance of the interior and boundary functions in the minimization process.

The integrals of the cost function are often labeled as:

$$L_I(v) \equiv \int_\Omega |\, \mathcal{N}_I(v(t,x)) \,|^p \, dxdt, \tag{3.32}$$

and

$$L_B(v) \equiv \int_{\partial\Omega} (\mid \mathcal{N}_B(v(t,x)) \mid^p + \mid \mathcal{N}_0(v(t,x)) \mid^p) \, dxdt, \tag{3.33}$$

which denote the interior and the boundary cost functions respectively.

3.1.3.4. Integration: Fourier based methods

Fourier techniques can be applied in the context of the risk-neutral valuation framework by solving the the integral in Fourier space. Fourier techniques rely on the availability of the *Characteristic function* (ChF), i.e., the Fourier transform of the PDF associated to the asset stochastic process. They form the so-called *Fourier pair*, as

$$\begin{aligned} \hat{f}(u) &= \int_{\mathbb{R}} \exp(ixu) f(x) dx, \\ f(x) &= \frac{1}{2\pi} \int_{\mathbb{R}} \exp(-ixu) \hat{f}(u) du, \end{aligned} \tag{3.34}$$

where $\hat{f}$ denotes a Fourier transformed function, i.e., here, the ChF.

Contrary to the PDF, the ChF can be derived for many models in finance, particularly those that are driven by Lévy processes[5]. Therefore, the PDF can be recovered from the ChF via Fourier inversion. Thus, the PDF can be directly employed within the risk-neutral valuation formula in Equation (3.9). The starting point is the representation of the unknown PDF in the form of an *orthonormal* series expansion on a finite interval $[a, b]$, $z \in [a, b]$, as

$$f(z) = \frac{1}{b-a} \left(1 + 2 \sum_{k=1}^{\infty} A_k \cdot \psi_k(z) \right), \tag{3.35}$$

where $\psi_k(\cdot)$ are orthonormal basis functions and the expansion coefficients A_k can be obtained by

$$A_k := \langle f, \psi_k \rangle = \int_a^b f(z) \psi_k(z) dz. \tag{3.36}$$

By using Parseval's identity, i.e., $\langle f, \psi_k \rangle = \frac{1}{2\pi} \langle \hat{f}, \hat{\psi}_k \rangle$, where $\hat{f}$ and $\hat{\psi}_k$ denote the Fourier transformations of f and ψ_k, respectively, the expansion coefficients can be computed based on the ChF,

$$A_k := \frac{1}{2\pi} \langle \hat{f}, \hat{\psi}_k \rangle = \frac{1}{2\pi} \int_a^b \hat{f}(u) \hat{\psi}_k(u) du. \tag{3.37}$$

Fourier inversion methods usually provide a high accuracy at limited computational cost. However, their applicability strongly relies on the availability of the ChF. In some cases when a ChF is not directly available, the ChF can be approximated, but this may hamper the efficiency of the methodology. The curse of dimensionality in the case of multi-dimensional problems will also be problematic when Fourier techniques are employed.

Several pricing techniques have been developed based on a Fourier approach, like the Carr and Madan method (Carr & Madan, 1999), Fang and Oosterlee (Fang & Oosterlee, 2008) and Ortiz-Gracia and Oosterlee (Ortiz-Gracia & Oosterlee, 2016).

3.1.3.5. Trinomial trees

The trinomial tree scheme allows us to simulate the SDE (3.5) under the risk neutral measure (Kubo et al., 2020). In order to do so, we discretize the time as $t_j = j\Delta t$ with $j = 0, 1, ..., J$ and the underlying as $s_i = i\Delta s$ with $i = 0, 1, ..., I$. Here $t_J = T$ and $s_I = s_\infty$.

[5] A general class of processes with independent and stationary increments and continuous in probability. A Brownian motion is an example of Lévy processes where the increments follow a Gaussian distribution.

Next, we define the transition probabilities as:

$$\begin{aligned} p_u(s,t) &= Prob[S_{t+\Delta t} = s + \Delta s | S_t = s], \\ p_m(s,t) &= Prob[S_{t+\Delta t} = s | S_t = s], \\ p_d(s,t) &= Prob[S_{t+\Delta t} = s - \Delta s | S_t = s]. \end{aligned} \tag{3.38}$$

To approximate the SDE we choose the transition probabilities between the nodes so that they reproduce the first and the second moments of the process:

$$\begin{aligned} p_u(s_i,t_j) &= \frac{1}{2}\left(\frac{\sigma^2 s_i^2}{\Delta s^2} + \frac{r s_i}{\Delta s}\right)\Delta t, \\ p_d(s_i,t_j) &= \frac{1}{2}\left(\frac{\sigma^2 s_i^2}{\Delta s^2} - \frac{r s_i}{\Delta s}\right)\Delta t, \\ p_m(s_i,t_j) &= 1 - p_u - p_d. \end{aligned} \tag{3.39}$$

These probability transitions allows us to compute the final distribution of the asset $f(s_T|\mathcal{F}_t)$. To compute the value of an option we then need to use equation (3.9).

3.2. Risk measures for derivatives portfolios

Pricing and computing risks of financial derivatives are linked problems that share core parts of the formulation and resolution techniques, but with important differences:

- Probability measure:

  [Pricing]: Risk neutral probability measure Q is used. In fact, model parameters are calibrated so that they match the prices of standard options observed in the market; this is called *market implied calibration*.

  [Risks]: the problems are formulated under *real probability* P. The real probability usually is approximated by the historical probability.

  For instance in the case of parametric models, as Black-Scholes, parameters are calibrated so that they match the statistical features observed in the historical time series (*historical calibration*).

- Measure:

  [Pricing]: Computing expectations of a distribution.

  [Risks]: Computing tails of a distribution, as quantiles.

- Single trade versus portfolio:

  [Pricing]: Prices are computed individually at trade level [6].

  [Risks]: It is important to consider portfolios of derivatives, so that netting/hedging effects are taken into account. Linked to this, very often risk problems are usually highly dimensional.

For the sake of simplicity, we will focus on a couple of important financial risks, market risk and credit risk. A complete and comprehensive monograph on risk management can be found in (McNeil et al., 2015).

3.2.1. Market risk

Market risk focuses into the losses of a portfolio during a short time interval, due to shocks/movements on the underlying risk factors.

Let $V_P(t, S_t^1, \ldots, S_t^d)$ denote the value at time t of a derivatives portfolio that depends on the underlying assets $S^1, \ldots, S^d$.

[6] Except for the XVA adjustments.

We first define the portfolio value increment during $[t, t + \Delta t]$

$$\Delta V_P = V_P(t, S^1_{t+\Delta t}, \ldots, S^d_{t+\Delta t}) - V_P(t, S^1_t, \ldots, S^n_t), \tag{3.40}$$

where $S^i_{t+\Delta t}$ is the underlying asset scenario at $t + \Delta t$. Most common measures do not take into account portfolio ageing, that is why in the definition the underlyings are shocked while the time is fixed.

From the portfolio value increments, the portfolio losses are given by

$$L := -\Delta V_P \tag{3.41}$$

Notice that, given a distribution of underlying scenarios, a distribution of portfolio value increments, and therefore a distribution of losses, can be derived.

3.2.1.1. Delta-Gamma approximation

Although simple in definition, the practical computation of the typical risk measures is a challenging and computationally expensive problem, especially when the changes in V_P cannot be assumed linear in S. Then, the estimation of such a risk measures is often performed by means of an Monte Carlo method. In order to find a balance between accuracy and tractability, one of the most employed methodologies is the so-called Delta-Gamma approximation which combines Monte Carlo path generation (or any other scenario generation), a second-order Taylor expansion and the sensitivities (derivatives) to reduce the computational cost and capture the non-linearity in portfolio changes. The delta-gamma approximation of ΔV_P (in the case of only one risk factor) is given by

$$\Delta V_P \approx \sum_{i=1}^{I} w_i \frac{\partial v_i}{\partial S} \Delta S + \frac{1}{2} \sum_{i=1}^{I} w_i \frac{\partial^2 v_i}{\partial S^2} (\Delta S)^2,$$

with I the number of assets depending on risk factor S, w_i and v_i the amount and the value of asset i, respectively. The partial derivatives are evaluated at initial time t. In the case of options contracts, these partial derivatives correspond to the $\Delta = \partial V_P / \partial S$ and $\Gamma = \partial^2 V_P / \partial S^2$ sensitivities.

We can again use a parametric model assuming that the distribution of ΔS is known (normal, Student's $\mathbf{t}$, etc); but non parametric alternatives are as well used in practice.

3.2.1.2. Underlying scenario computation

For computing $S^i_{t+\Delta t}$ in (3.40), or equivalently, ΔS for the Delta-Gamma approximation, different alternatives can be considered:

- A parametric model for S, like Black-Scholes, calibrated to historical data; or directly a parametric distribution of ΔS like normal, Student's $\mathbf{t}$, etc
- (Non parametric) Historical *multiplicative shocks* as:

  $$S^i_{t+\Delta t} = S^i_t \; R^i_{\Delta t}$$

  with

  $$R^i_{\Delta t} = S^i_s / S^i_{s-\Delta t}$$

  for $s < t$ an observed historical date.
- Other non parametric approach.

In the following we will be defining specific market risk measures that are no more than statistical measures of the distribution of L, focusing on the upper tail.

3.2.2. Classical risk measures: VaR and CVaR

We consider two well-known measures of risk, the *Value-at-Risk* (VaR) and the *Conditional Value-at-Risk* (CVaR) (also known as *Expected Shortfall*). We introduce the definition of both measures in the following results.

Definition 1. *Given a confidence level $\alpha \in (0,1)$, the portfolio VaR is defined as,*

$$VaR_\alpha = \inf\{l \in \mathbb{R} : \mathbb{P}(L \leq l) \geq \alpha\} = \inf\{l \in \mathbb{R} : F_L(l) \geq \alpha\},$$

where F_L is the distribution function of the total loss random variable L (we emphasize the dependence of VaR with respect to the risk exposures).

The VaR is therefore an estimate of how much one can lose from one's portfolio over a given time horizon, with a given degree of confidence (Wilmott, 2007).

Definition 2. *Given the loss variable L with $\mathbb{E}[|L|] < \infty$ and distribution function F_L, the CVaR at confidence level $\alpha \in (0,1)$ is defined as,*

$$CVaR_\alpha = \frac{1}{1-\alpha}\int_\alpha^1 VaR_u du.$$

When the loss variable is integrable with continuous distribution function, then the CVaR satisfies the equation,

$$\text{CVaR}_\alpha = \mathbb{E}[L|L \geq \text{VaR}_\alpha], \tag{3.42}$$

or, in integral form,

$$\text{CVaR}_\alpha = \frac{1}{1-\alpha}\int_{\text{VaR}_\alpha}^{+\infty} x f_L(x) dx, \tag{3.43}$$

where f_L is the probability density function of the total loss random variable L. Thus, in the continuous case, CVaR can be interpreted as the expected loss in the event that VaR is exceeded.

3.2.3. Credit Portfolio Management

Another source of risk, that needs to be measured and managed is the credit risk coming from the risk of default of a number of obligors. The financial instruments could be simple loans or bonds, or more complex products such as credit derivatives such as Credit Default Swaps or Credit Loan Obligations. It can also include credit exposures stemming from other derivative contracts through counterparty credit risk. Given its complexity and high dimensionality, credit portfolio modelling relies on dimensionality reduction techniques such as factor models and Principal Component Analysis (PCA).

Let us consider a portfolio of N credit obligations (e.g. loans, bonds, overdrafts, etc) to a number of obligors or counterparties. Each instrument is characterized by the *exposure at default* and the *loss given default*. The exposure of default is the amount of money that would be due at the time of counterparty default. The loss given default is the actual loss incurred after the recovery process is concluded (this is bounded between zero and 100%). In addition, each counterparty has a known *probability of default* associated to each credit worthiness. Note that several instruments can be linked to the same counterparty, however here we will take the simplifying assumption that each counterparty has only one single instrument. While the first two parameters will be denoted by E_j and P_j, $j = 1, \ldots, J$, the third parameter is assumed to be 100% for all the obligors. These parameters can be estimated from the capital markets, or from historical credit data.

Assume now that we are in the framework of Merton's firm-value model (Merton, 1974). In this approach, a counterparty is assumed to go into default if the value of its assets fall below a given default barrier which is linked to the value of its liabilities. In other words, if the value of the assets of a company falls bellow how much the company owns to its creditors, the model assumes that this firm is in default. Therefore, the combination of asset value and liability barrier determines determines the credit quality of the obligor and defines its *probability of default*. Let $V_j(t)$ denote the asset value of instrument j at time $t < T$, where T is the time horizon (typically one year). The obligor j defaults when its value at the end of the observation

period, $V_j(T)$, falls below barrier , τ_j, i.e, $V_j(T) < \tau_j$. A default indicator can be defined mathematically as

$$D_j = \mathbb{1}_{\{V_j(T)<\tau_j\}} \sim Be\left(\mathbb{P}\left(V_j(T) < \tau_j\right)\right),$$

where $Be(p)$ is a Bernoulli distribution with probability of success p. Given D_j, the individual loss of obligor j is defined as,

$$L_j = D_j \cdot E_j,$$

while the total loss in the portfolio reads,

$$L = \sum_{j=1}^{J} L_j. \tag{3.44}$$

Because the credit quality of the obligors (captured by the asset value in the Merton model) is correlated, this problem is closely related to the basket option covered earlier, but with the additional complexity of having different barriers. In addition, banks portfolios would usually comprise several millions of obligors, and therefore Credit Portfolio models typically relay on Monte Carlo techniques to estimate the probability distribution of portfolio losses. Once the probability of portfolio losses L has been estimated, it is possible to compute different risk measures such as those covered later in Section 3.2.2.

Furthermore, the credit portfolio model described above captures losses from default events only (these types of models are typically called default/no-default models). In reality, credit portfolio models also take into account losses arising from changes in credit quality and that impact the bank's profit and loss accounts (e.g. credit migration for fair valued assets and stage migration under IFRS 9 accounting). In this models, the probability of migration needs to be considered in addition to the probability of default, for each obligor the problem changes from the binomial (default/no-default) approach considered above to a multi-nominal problem.

3.2.3.1. Factor models for portfolio credit risk

A common approach used to reduced the dimensional of the problem is the introduction of Factor models. In this models the credit quality of each obligor is assumed to be driven by two different components: a set of factors that is shared by the the obligors and captures the systematic risk and a second that is unique to each obligor and capture its idiosyncratic risk. Depending on the number of factors of the systematic part, the model can be classified into the *one-* or *multi-factor* class. The power of factor models is that allows us to reduce the number of parameters needed to capture portfolio correlation to be estimated in the model from order J^2, i.e. pair-wise correlations between each obligor in the portfolio, to one of order J. In the following section we briefly describe some of the most commonly employed models.

One-factor models

In the one-factor model setting, the credit quality (which in Merton model is defined as the logarithmic return of the asset value) of obligor j, X_j, at time T is represented by a common, standard normally distributed single factor Y component and an idiosyncratic Gaussian noise component ε_j. The dependence structure between these two latent random variables can be set using copula functions. Thus, these models are also called one-factor copula models. Two models are usually considered in practice. The *Gaussian copula model* is given by:

$$X_j = \sqrt{\rho_j} Y + \sqrt{1-\rho_j}\varepsilon_j, \tag{3.45}$$

where Y and ε_j are i.i.d. standard normal random variables for all $j = 1, \ldots, J$. Alternatively, as an extension of the model in Equation (3.45), the *t-copula model* was introduced to take into account tail dependence,

$$X_j = \sqrt{\frac{\nu}{W}}\left(\sqrt{\rho_j} Y + \sqrt{1-\rho_j}\varepsilon_j\right), \tag{3.46}$$

where $\varepsilon_1, \cdots, \varepsilon_J, Y \sim \mathcal{N}(0,1)$, W follows a chi-square distribution $\chi^2(\nu)$ with ν degrees of freedom and $\varepsilon_1, \cdots, \varepsilon_J$, Y and W are mutually independent. Scaling the model in Equation (3.45) by the factor $\sqrt{\nu/W}$ transforms standard Gaussian random variables into t-distributed random variables with ν degrees of freedom. For both models, the parameters $\rho_1, \cdots, \rho_J \in (0,1)$ are the correlation coefficients. In case that

$\rho_j = \rho$, for all $j = 1, \ldots, J$, the parameter ρ is called the common asset correlation.

According to the Merton's model described above, obligor j defaults when the value of its assets falls below the barrier τ_j. The barrier is therefore defined by $\tau_j := \Phi^{-1}(P_j)$ or $\tau_j := \Phi_\nu^{-1}(P_j)$ for the Gaussian and t-copula models respectively, where Φ^{-1} denotes the inverse of the standard normal cumulative distribution function and Φ_ν^{-1} is the corresponding inverse distribution function of the t-distribution (with ν degrees of freedom).

Multi-factor models

Multi-factor models aim to capture more realistic correlation structures, e.g. obligors in similar industry sector and geographies would typically be more correlated. For this, we consider the extension to multiple dimensions of the models presented in Section 3.2.3.1, i.e., the *multi-factor Gaussian copula model* and the *multi-factor t-copula model*.

The d-factor Gaussian copula model assumes that the covariance structure of $[V_1, \ldots, V_J]$ is determined by the multi-factor model,

$$X_j = \boldsymbol{a_j}^T \boldsymbol{Y} + b_j \varepsilon_j, \quad j = 1, \cdots, J. \tag{3.47}$$

where $\boldsymbol{Y} = [Y_1,\ Y_2, \ldots, Y_d]^T$ denotes the systematic risk factors. Note that we represent vectors by bold symbols throughout this report. Here, $\boldsymbol{a_j} = [a_{j1}, a_{j2}, \ldots, a_{jd}]^T$ represents the factor loadings satisfying $\boldsymbol{a_j}^T \boldsymbol{a_j} < 1$, and ε_j are standard normally distributed random variables representing the idiosyncratic risks, independent of each other and independent of $\boldsymbol{Y}$. The constant b_j, being the factor loading of the idiosyncratic risk factor, is chosen so that V_j has unit variance, i.e., $b_j = \sqrt{1 - \left(a_{j1}^2 + a_{j2}^2 + \cdots + a_{jd}^2\right)}$, which ensures that V_j is $\mathcal{N}(0, 1)$.

The incentive for considering the multi-factor version of the Gaussian copula model becomes clear when one rewrites it in matrix form,

$$\begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_J \end{bmatrix} = \begin{bmatrix} a_{11} \\ a_{21} \\ \vdots \\ a_{J1} \end{bmatrix} Y_1 + \begin{bmatrix} a_{12} \\ a_{22} \\ \vdots \\ a_{J2} \end{bmatrix} Y_2 + \cdots + \begin{bmatrix} a_{1d} \\ a_{2d} \\ \vdots \\ a_{Jd} \end{bmatrix} Y_d + \begin{bmatrix} b_1\varepsilon_1 \\ b_2\varepsilon_2 \\ \vdots \\ b_J\varepsilon_J \end{bmatrix}.$$

While each ε_j represents the idiosyncratic factor affecting only obligor j, the common factors $Y_1, Y_2 \ldots, Y_d$, may affect all (or a certain group of) obligors. Although the systematic factors are sometimes given economic interpretations (as industry or regional risk factors, for example), their key role being that they allow us to model complicated correlation structures in a non-homogeneous portfolio.

Similarly, the multi-factor t-copula model definition reads,

$$X_j = \sqrt{\frac{\nu}{W}} \left(\boldsymbol{a_j}^T \boldsymbol{Y} + b_j \varepsilon_j\right), \quad j = 1, \cdots, J, \tag{3.48}$$

where $\boldsymbol{Y}, \varepsilon_j, \boldsymbol{a_j}$ and b_j are defined as before, with $W \sim \chi^2(\nu)$.

3.2.4. Numerical methods

3.2.4.1. Monte Carlo

Assuming the loss distribution as defined in Equation (3.44) and employing Monte Carlo methods, the VaR can be computed as follows:

- Generate M samples of L, denoted by $L^1, L^2, \ldots, L^M$.
- Compute the empirical CDF:

$$\tilde{F}_{L,M}(x) = \frac{1}{M} \sum_{m=1}^{M} \mathbb{1}_{\{L^m \leq x\}}.$$

- Then, we have the estimator,

$$\mathrm{VaR}_\alpha \approx \tilde{F}^{-1}_{L,M}(\alpha).$$

Practically, the same result can be achieved by first sorting the sample set, obtaining the ordered samples $L^{\bar{1}}, L^{\bar{2}}, \ldots, L^{\bar{M}}$ and taking

$$\mathrm{VaR}_\alpha \approx \min_{L^{\bar{m}}} \left\{ \bar{m} \geq \lfloor \alpha M \rfloor \right\},$$

where $\lfloor \cdot \rfloor$ denotes the nearest integer smaller than the argument.

Given the VaR value, a Monte Carlo estimator for the CVaR can be readily derived,

$$\mathrm{CVaR}_\alpha \approx \frac{\sum_{m=1}^{M} \mathbb{1}_{\{L^{\bar{m}} \geq \mathrm{VaR}_\alpha\}} L^{\bar{m}}}{\sum_{m=1}^{M} \mathbb{1}_{\{L^{\bar{m}} \geq \mathrm{VaR}_\alpha\}}}.$$

Notice that when sampling L, parametric or non parametric distributions can be used.

The VaR (and the CVaR) are intended to prevent extreme events of big losses, so the quantile α is usually significantly high, between 95% and 99.9% depending on the application. In such regimes, Monte Carlo is rather inefficient, specially for the CVaR computation, since the number of samples in the area of interest is usually not sufficient to provide an acceptable precision. In order to mitigate this drawback, several approaches have been explored in the literature, being the utilization of *importance sampling* techniques one of the most successful attempts.

3.2.4.2. Other methodologies based on the integral formulation

Taking advantage of the integral formulation in Equation 3.43, other numerical methodologies can be explored in this context. For example, in a similar way as for option pricing, the density function f_L can be approximated by an expansion of orthogonal basis functions, which opens the alternative of employing, for example, Fourier inversion methods. Another approach is to directly approximate the distribution function F_L, which would allow to derive closed-form expressions to compute the VaR and the CVaR values.

3.2.4.3. Principal Component Analysis

Principal Component Analysis is a classical mathematical technique that is widely used in quantitative finance for dimensionality reduction; not only for both pricing problems (as Section 3.1.1.3)) and in risk models (as the ones in Section 3.2.3.1), but also for stock value forecasting (usually in combination with other machine learning techniques).

As we have seen in equation (3.6), a multidimensional problem with N risk factors can be formulated through a $N \times N$ correlation matrix (ρij), and a N volatility vector (σ_i). Equivalently, it can be formulated through the Hermitian and positive semi-definite covariance matrix Σ, defined as

$$\Sigma_{ij} = \rho_{ij}\sigma_i\sigma_j$$

The objective is to reduce the dimension of the problem from N to E while still representing the highest amount of variance. For this:

- First Σ is factorized in terms of its eigenvalues and eigenvectors through the *spectral decomposition*.
- Then, only the highest (with higher eigenvalues) E ($E < N$) components are kept, and the $N - E$ smallest are discarded.

4. Quantum toolkit for finance

In Chapter 3 we have discussed some of the most widely used techniques in classical finance. Throughout this chapter we will cover some proposals for their quantum counterparts.
In the first section we will be covering integration techniques (sometimes referred to as Quantum Computing Monte Carlo (QCMC)[1]).
In the second section we will be covering alternative quantum algorithms to the ones discussed in the first section which also have an immediate application to finance.
Finally, in the third section we will discuss some of the challenges and promising research lines emerging on the subject.

4.1. Quantum alternatives to Monte Carlo Algorithms for Applications in Finance

In this section, we briefly revise the state of the art of the quantum alternatives to Monte Carlo algorithms for the numerical evaluation of expectation values. Then, we describe some concrete applications to the financial problem of pricing European vanilla options, as introduced in Section 3.1.1.1, and comment about the computation of risk indicators (see Section 3.2.2).

4.1.1. Introduction

The possibility of handling Hilbert spaces which grow exponentially in the number of qubits makes quantum computers an attractive framework where to address function integration over large, multi-dimensional domains. Such exponential growth of the dimension of the Hilbert space suggests that the quantum alternative can be in a better position with respect to its classical counterpart in dealing with combinatorial explosions and the curse of dimensionality. However, bringing this hope to practice encounters several bottlenecks, especially in relation to current or near-term quantum technologies.

We will discuss the state of the art of quantum numerical integration examining the most promising integration algorithms described in the literature and their implementation. They all belong to the same family originated from Grover's search algorithm (L. K. Grover, 1996). As such, they are general-purpose and entail a theoretical quadratic speed-up. This means that they are not based on any particularly restrictive assumption on the function to be integrated, thereby encompassing quite generic cases, lying in this regard on a similar footing as classical Monte Carlo algorithms. In particular, we focus on the problem of the numerical evaluation of expectation values, where the integrand function results from the multiplication of a probability density distribution and the function whose expected value we want to get.

The quadratic speed-up of quantum algorithms with respect to their classical counterparts means that, if the precision for classical Monte Carlo sampling generically scales as $M^{-\frac{1}{2}}$, where M is the number of samples, the quantum algorithms have a theoretical precision scaling as M^{-1}. Roughly, this can be motivated by the fact that quantum algorithms work at the level of the probability amplitude instead of the probability, thus gaining a quadratic factor. This intuitive description will be made more precise below.

Figure 2 shows a schematic pipeline of a quantum integration algorithm for the computation of the expected value of a function $f(x)$ with respect to a Probability Density Function $p(x)$. The processing is divided in four blocks associated respectively to the **loading of** $p(x)$, the **loading of the function** $f(x)$, a **quantum amplification** part (relying on Grover's amplification strategy) and an **estimation** part which can be performed in different ways, such as an inverse quantum Fourier transform or approximate counting strategies, possibly followed by classical post-processing.

To avoid unnecessary complications, and to keep the treatment generic, we consider that x spans an unspecified metric space and we assume to have a procedure (or an oracle) which provides a suitable discretization, that is able to translate the desired integral into a discrete sum. We henceforth overload the notations: the symbol x will run over the discretized domain, and the functions $p(x)$ and $f(x)$ are the appropriate discretized versions of the original functions (possibly taking into account non-trivial measures).

[1] This should not be confused with the general concept of Quantum Monte Carlo, a set of computational methods to calculate complex quantum systems

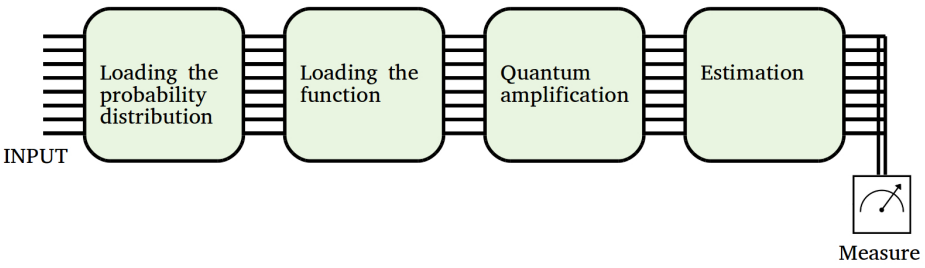


***Figure 2**: Schematic pipeline of a quantum algorithm to compute the expected value of a function*

We stress that the discretization procedure is a relevant aspect of the computation which can introduce approximations, this is however common to classical and quantum algorithms.[2] The discretization process can have some relation to that of the loading of the initial quantum state, see Section 4.3.1 for further comments on this issue.

4.1.2. Quantum amplitude amplification and estimation

In order to describe the algorithm of quantum amplitude amplification and estimation (see Fig. 2), we borrow from the abstract of (Brassard et al., 2000), where the algorithm was introduced. We first describe the amplification part and later the estimation part.

Let us take a Boolean function $\chi : X \to \{0, 1\}$ which partitions the set X labelling the elements as *good* when $\chi(x) = 1$ and as *bad* when $\chi(x) = 0$. Let us take also a generic quantum algorithm $\mathcal{A}$ (without measurements) such that

$$\mathcal{A}|0\rangle = \sum_{x \in X} \alpha_x |x\rangle \, . \tag{4.1}$$

Let us denote with a the probability that an element marked as *good* is measured on $\mathcal{A}|0\rangle$. If one repeats the process of taking $|0\rangle$, applying $\mathcal{A}$ to it and then measuring, one expects to have to repeat the procedure on average $\frac{1}{a}$ times before encountering a marked element. Instead, by means of *amplitude amplification*, the expected number of times the experiment needs to be repeated is proportional to $\frac{1}{\sqrt{a}}$. Therefore, we get a quadratic speed-up.

4.1.2.1. Grover-like amplification

In order to appreciate how the *amplitude amplification* algorithm works we need to enter into its details.

Let us consider a real function f defined on a discrete set of points labelled in binary notation. As already commented, this represents a discretized domain where the individual discrete points have a label given by a binary number. Note that the domain can be multi-dimensional, and we simply give binary "proper names" (or addresses) to each discrete point. As pointed at the beginning of this section, the function f defined on the discrete domain can be thought of as a discrete approximation of a more generic function. For technical reasons that will become evident soon, the function f is required to take values within the real interval $[0, 1]$. If needed, we can define f by means of a rescaling of the actual function that we want to integrate and then rescale back at the end of the computation, exploiting the linearity of integration. Hence, there is no loss of generality in the assumption.[3]

Thus, we have a function f

$$f : x \in X = \{0, 1\}^n \to [0, 1] \, , \tag{4.2}$$

and our purpose is to compute the sum[4]

[2] A simple example of discretization arises when using a fixed point arithmetic in classical computers.
[3] It is however important to recall that such rescaling affects the final estimation of errors.
[4] Recall that, upon the discretization procedure (which we are not specifying to keep the treatment general), the sum in (4.5) corresponds to the discretized version of the original integral we needed to compute. The problem of defining a suitable (and efficient) discretization procedure, common to classical and quantum algorithms, entails the theory of measure and the choice of optimal

$$\sum_{x\in X} f(x) \ . \tag{4.5}$$

We can immediately generalize the problem of integration to that of computing expectation values over a generic probability distribution $p(x)$, namely, that of computing expressions like

$$\sum_{x\in X} p(x)f(x) \ . \tag{4.6}$$

Even if (4.6) can be rightly regarded as just an integral of the function $p(x)f(x)$, there are technical reasons to factor explicitly the probability distribution. We will return to the point later. Let us just recall that, being $p(x)$ a normalized probability distribution, we have

$$\sum_{x\in X} p(x) = 1 \ . \tag{4.7}$$

We assume to be able to implement two operators, $\mathcal{P}$ and $\mathcal{R}$, which respectively "load" the probability density distribution $p(x)$ and the function $f(x)$ to the quantum state respectively. More precisely, their action is specified by

$$\mathcal{P}|0\rangle_n = \sum_{x=0}^{2^n-1} \sqrt{p(x)}\ |x\rangle_n \ , \tag{4.8}$$

and

$$\mathcal{R}|x\rangle_n|0\rangle = |x\rangle_n \left(\sqrt{f(x)}|1\rangle + \sqrt{1-f(x)}|0\rangle\right) \ . \tag{4.9}$$

We have indicated with $|x\rangle_n$ the quantum register containing the states associated to the discretization of the domain[5], the label x running over their binary address; n denotes the number of qubits in such register. In (4.9) there is an extra auxiliary qubit which represents a "flag" denoting the "good" states whose amplitude corresponds to the function we wanted to load. The operator $\mathcal{R}$ can be implemented by means of rotations controlled by the physical register.

Composing (4.9) and (4.8), we obtain

$$|\Psi\rangle \equiv \mathcal{R}\left(\mathcal{P}\otimes\mathbb{1}\right)|0\rangle_n|0\rangle = \sum_{x=0}^{2^n-1} \sqrt{p(x)}\ |x\rangle_n \left(\sqrt{f(x)}|1\rangle + \sqrt{1-f(x)}|0\rangle\right) \ . \tag{4.10}$$

The probability of getting the auxiliary qubit equal to 1 in a measurement of the state $|\Psi\rangle$ is given by

$$\mathbb{E}_{x\sim p}\left(f\right) = \sum_{x=0}^{2^n-1} p(x)f(x) \ , \tag{4.11}$$

which is actually the expectation value we want to compute. So far, however, apart from the quantum implementation, we have no gain: computing $\mathbb{E}_{x\sim p}\left(f\right)$ by repeating the preparation and measurement of $|\Psi\rangle$ brings no advantage with respect to a classical sampling.

The next steps is where the Grover strategy kicks in, leading to a quantum advantage. The algorithm ex-

mashes. To avoid being too abstract, let us make an explicit example. Consider integrating the function $\sin(x)$ over $[0,\pi]$,

$$I = \int_0^\pi dx \sin(x) \ . \tag{4.3}$$

We can choose a uniform discretization $x \in \{n\pi/N\}$ for $n = 1, ..., N$ with N some large integer. We have

$$I \sim \frac{\pi}{N}\sum_n \sin(n\pi/N) \ . \tag{4.4}$$

The coefficient $\frac{\pi}{N}$ corresponds to the measure of the discretized intervals and it is important to take it into account in defining the discretized version of the original function.

[5] Referred to as the *physical register*, as opposed to the *auxiliary registers*.

ploits similar ideas to the original Grover search algorithm.[6] In particular, we want a controlled enhancement (or *amplification*) of the probability of getting "good" states by measuring the quantum state. This will make the integration more efficient.

The path to obtain the *amplification* consists in some specific manipulations detailed below. First, one defines the following vectors:

$$|\Psi_1\rangle = \sum_{x=0}^{2^n-1} \sqrt{p(x)f(x)}\, |x\rangle_n |1\rangle \,, \tag{4.14}$$

$$|\Psi_0\rangle = \sum_{x=0}^{2^n-1} \sqrt{p(x)\,(1-f(x))}\, |x\rangle_n |0\rangle \quad . \tag{4.15}$$

The vectors $|\Psi_1\rangle$ and $|\Psi_0\rangle$ belong to the 2^{n+1} dimensional Hilbert space $\mathcal{H}$. This latter is generated by the collection of basis states $\{|x\rangle_n|1\rangle, |x\rangle_n|0\rangle\}$ for all values of x (we have 2^n such values). More precisely, $|\Psi_1\rangle$ and $|\Psi_0\rangle$ indicate two specific directions belonging respectively to the subspaces of $\mathcal{H}$ associated to the values 1 and 0 of the auxiliary qubit. Said otherwise, $|\Psi_1\rangle$ and $|\Psi_0\rangle$ generate a plane $\pi \subset \mathcal{H}$.

Notice that, from the definitions (4.14) and (4.15), we have

$$|\Psi\rangle = |\Psi_1\rangle + |\Psi_0\rangle \,, \tag{4.16}$$

so the state $|\Psi\rangle$ belongs to the plane π. One can also define

$$a \equiv \langle\Psi|\Psi_1\rangle = \langle\Psi_1|\Psi_1\rangle = \sum_{x=0}^{2^n-1} p(x)f(x) \,, \tag{4.17}$$

$$1-a \equiv \langle\Psi|\Psi_0\rangle = \langle\Psi_0|\Psi_0\rangle = \sum_{x=0}^{2^n-1} p(x)\,(1-f(x)) \quad , \tag{4.18}$$

and we remind ourselves that getting an estimation for a, that is the expectation value of f, is our purpose. For the sake of subsequent manipulations, let us also define the normalized version of (4.14) and (4.15), namely

$$|\tilde{\Psi}_1\rangle = \frac{1}{\sqrt{a}} \sum_{x=0}^{2^n-1} \sqrt{p(x)f(x)}\, |x\rangle_n |1\rangle \,, \tag{4.19}$$

$$|\tilde{\Psi}_0\rangle = \frac{1}{\sqrt{1-a}} \sum_{x=0}^{2^n-1} \sqrt{p(x)\,(1-f(x))}\, |x\rangle_n |0\rangle \,, \tag{4.20}$$

thus, we have

$$|\Psi\rangle = \sqrt{a}\, |\tilde{\Psi}_1\rangle + \sqrt{1-a}\, |\tilde{\Psi}_0\rangle \,. \tag{4.21}$$

The core idea of the amplification algorithm consists in rotating the state $|\Psi\rangle$ in the plane π (spanned by the vectors $|\tilde{\Psi}_0\rangle$ and $|\tilde{\Psi}_1\rangle$) in order to enhance *in a controlled way* the amplitude of measuring $|\tilde{\Psi}_1\rangle$. For this

[6] The original Grover algorithm to search a marked element $\bar{x}$ chooses

$$p(x) = \frac{1}{N} \,, \tag{4.12}$$

which is the uniform sampling (implemented through a Walsh-Hadamard transform) and the oracle is

$$f(x) = \delta_{x\bar{x}} \,, \tag{4.13}$$

where δ is a Kronecker delta.

purpose, we define the Grover amplification operator[7]

$$Q \equiv U_\Psi U_{\tilde{\Psi}_0} \,, \tag{4.24}$$

where

$$U_\Psi \equiv \mathbb{1} - 2|\Psi\rangle\langle\Psi| \,, \tag{4.25}$$

$$U_{\tilde{\Psi}_0} \equiv \mathbb{1} - \frac{2}{1-a}|\Psi_0\rangle\langle\Psi_0| = \mathbb{1} - 2|\tilde{\Psi}_0\rangle\langle\tilde{\Psi}_0| \,, \tag{4.26}$$

are specific reflection operators. Indeed, U_Ψ and $U_{\tilde{\Psi}_0}$, when applied to a generic quantum state, flip its component along $|\Psi\rangle$ and $|\tilde{\Psi}_0\rangle$, respectively.

Let us obtain an explicit expressions for the operators U_Ψ and $U_{\tilde{\Psi}_0}$ introduced in (4.25) and (4.26). For the latter, we have

$$U_{\tilde{\Psi}_0} = -U_{\tilde{\Psi}_1} = |\tilde{\Psi}_1\rangle\langle\tilde{\Psi}_1| - |\tilde{\Psi}_0\rangle\langle\tilde{\Psi}_0| = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \,, \tag{4.27}$$

where the last passage explicitly defines the meaning of the components of the 2×2 matrix in the right hand side.[8] In order to get an explicit expression for U_Ψ, it is convenient to define the angle θ through

$$\langle\Psi|\tilde{\Psi}_1\rangle = \sqrt{a} \equiv \sin\theta \,, \tag{4.29}$$

$$\langle\Psi|\tilde{\Psi}_0\rangle = \sqrt{1-a} = \cos\theta \,. \tag{4.30}$$

Then, taking into account that

$$\begin{aligned} |\Psi\rangle\langle\Psi| &= \left(\sqrt{a}|\tilde{\Psi}_1\rangle + \sqrt{1-a}|\tilde{\Psi}_0\rangle\right)\left(\sqrt{a}\langle\tilde{\Psi}_1| + \sqrt{1-a}\langle\tilde{\Psi}_0|\right) \\ &= \begin{pmatrix} a & \sqrt{a(1-a)} \\ \sqrt{a(1-a)} & 1-a \end{pmatrix} \\ &= \begin{pmatrix} \sin^2\theta & \sin\theta\cos\theta \\ \sin\theta\cos\theta & \cos^2\theta \end{pmatrix} \,, \end{aligned} \tag{4.31}$$

we have

$$U_\Psi = \begin{pmatrix} 1-2\sin^2\theta & -2\sin\theta\cos\theta \\ -2\sin\theta\cos\theta & 1-2\cos^2\theta \end{pmatrix} \,. \tag{4.32}$$

Combining the matrix expressions for U_Ψ and $U_{\tilde{\Psi}_0}$ given in (4.27) and (4.32), we can have an explicit expression for the Grover operator Q, namely

$$\begin{aligned} Q = U_\Psi U_{\tilde{\Psi}_0} &= \begin{pmatrix} 1-2\sin^2\theta & -2\sin\theta\cos\theta \\ 2\sin\theta\cos\theta & -1+2\cos^2\theta \end{pmatrix} \\ &= \begin{pmatrix} \cos^2\theta - \sin^2\theta & -2\sin\theta\cos\theta \\ 2\sin\theta\cos\theta & \cos^2\theta - \sin^2\theta \end{pmatrix} = \begin{pmatrix} \cos(2\theta) & -\sin(2\theta) \\ \sin(2\theta) & \cos(2\theta) \end{pmatrix} \,. \end{aligned} \tag{4.33}$$

Recall that from (4.29) and (4.30), the "initial state" $|\Psi\rangle$ is given by

$$|\Psi\rangle = \sin(\theta)|\tilde{\Psi}_1\rangle + \cos(\theta)|\tilde{\Psi}_0\rangle \,. \tag{4.34}$$

[7] Note that the Grover amplifier Q admits the following expression:

$$Q \equiv -\mathcal{A}S_0\mathcal{A}^{-1}S_f = U_\Psi U_{\tilde{\Psi}_0} \,, \tag{4.22}$$

where

$$\mathcal{A} \equiv \mathcal{R}\,(\mathcal{P}\otimes\mathbb{1}) \,. \tag{4.23}$$

The operator S_0 flips the sign of the $|0\rangle_n$ component; the operator S_f flips the sign of the states marked by $\chi = 1$, that is to say, it flips the sign of the "good" states.

[8] We rely on the completeness relation

$$|\tilde{\Psi}_1\rangle\langle\tilde{\Psi}_1| + |\tilde{\Psi}_0\rangle\langle\tilde{\Psi}_0| = \mathbb{1} \,. \tag{4.28}$$

Thus, we obtain

$$Q|\Psi\rangle = \sin(3\theta)|\tilde{\Psi}_1\rangle + \cos(3\theta)|\tilde{\Psi}_0\rangle \ . \tag{4.35}$$

Some observations are in order. First, the Grover operator Q actually implements a rotation in the π plane (generated by the state vectors $|\tilde{\Psi}_1\rangle$ and $|\tilde{\Psi}_0\rangle$); second, an iterated application of Q leads to

$$\mathcal{Q}^k|\Psi\rangle = \sin\big((2k+1)\theta\big)|\tilde{\Psi}_1\rangle + \cos\big((2k+1)\theta\big)|\tilde{\Psi}_0\rangle \ , \tag{4.36}$$

where k denotes a generic integer. If one chooses k in (4.36) such that $(2k+1)\theta \sim \frac{\pi}{2}$, then $\sin\big((2k+1)\theta\big) \sim 1$ and so one maximizes the probability of getting a result along $|\tilde{\Psi}_1\rangle$ upon measurement. We have therefore reached the initial purpose of *amplifying* the probability of measuring $|\tilde{\Psi}_1\rangle$, in a way which explicitly depends on θ (and thereby on a through (4.29)). Said otherwise, by measuring (4.36) we have a more efficient access to the estimation of a.

4.1.2.2. Amplitude estimation

Once we have the amplified state (4.36), we still need to actually estimate a in the best possible way. As already implicit in previous comments, (4.36) is the result of rotations due to the Grover operator Q which are aimed to enhance the probability of the quantum state vector to return $|\tilde{\Psi}_1\rangle$ upon measurement. So, a sampling by repeated preparations and measurements of (4.36), instead of the original vector $|\Psi\rangle$, already provides an advantage. However, there is room for an improvement in the precision of the estimation of a, and the path suggested in the original paper (Brassard et al., 2000) is to adopt an inverse quantum Fourier transform, see Figure 3. The first n qubits are the physical register upon which the block $\mathcal{P}$ loads the probability distribution. The block $\mathcal{R}$ applies the function whose expected value we want to compute, this is assumed to require one auxiliary qubit. The last m qubits are an auxiliary register used for amplitude estimation; it controls different powers of the Grover amplification block Q. Eventually, an inverse of the Quantum Fourier transform on the auxiliary register provides the phase estimation, from which one recovers the amplitude estimation.[9]

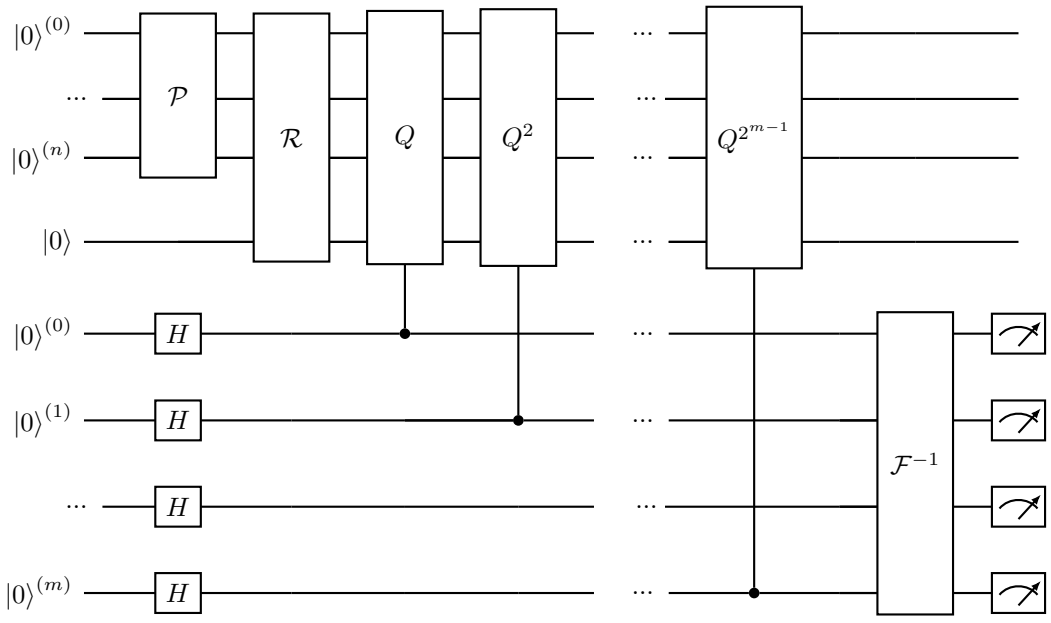


***Figure 3**: Quantum circuit for amplitude amplification and estimation (Brassard et al., 2000).*

[9] The paper (Montanaro, 2015) is generally regarded as representing the current state of the art in relation to quantum speedups in Monte Carlo tasks.

4.1.2.3. Interesting variations on the same theme

Amplitude estimation implemented by an inverse Quantum Fourier transform constitutes a bottleneck for the *amplitude amplification and estimation* algorithm (Brassard et al., 2000), especially in relation to current or near-future technologies due to the heaviness (*i.e.* required depth and length) of the associated quantum circuit. For this reason, some algorithms which need less resources have been proposed.

4.1.2.3.1. Quantum amplitude amplification and estimation with Max Likelihood One interesting possibility to avoid the inverse Quantum Fourier transform has been suggested in (Suzuki et al., 2020) and it relies on classical post-processing. One collects data corresponding to measuring states amplified by means of Q^k with different k (see Fig.4 for the circuit needed for a specific k); then one compares this dataset against a suitable classical prior distribution which depends on the angle θ (see (4.36)). Maximizing the likelihood that the distribution fits the data satisfactorily, provides an estimation of θ, here acting as a variational parameter. In turns, from the estimation of θ one gets an estimation of the amplitude a through (4.29).

The classical post-processing adopts standard statistical tools such as Fisher information and the Cramer-Rao inequality, we refer to (Cover, 2005) for their description. The Max likelihood alternative for amplitude estimation has been further discussed in subsequent papers. In (Grinko et al., 2020) the authors stress that the accuracy of the Max likelihood method (Suzuki et al., 2020) has not been precisely assessed, and they address this question in their appendix. The potential quadratic speedup of quantum amplitude estimation without quantum phase estimation has been covered in (Aaronson & Rall, 2020).

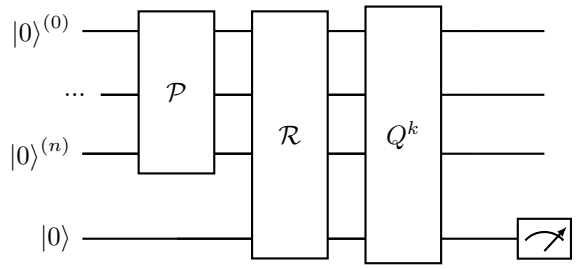


***Figure 4**: Quantum circuit for amplitude amplification and estimation with max likelihood (Suzuki et al., 2020). The circuit depicted refers to a specified value of the amplification exponent k. A collection of similar circuits for all the desired values of k is needed.*

4.1.2.3.2. Iterative quantum amplitude estimation In (Grinko et al., 2020) a different variant of quantum amplitude estimation is considered, which does not need quantum phase estimation, that is, it avoids the estimation of θ through an inverse Quantum Fourier transform. As such, the suggested implementation is able to reduce the number of qubits and gates, making the overall algorithm lighter. The analysis in (Grinko et al., 2020) focuses on a rigorous study of the quadratic quantum speedup for their algorithm.

To achieve this quadratic speed up, the iterative quantum amplitude estimation algorithm, like the other amplitude estimation algorithm, loads the square root of the integrand function in the quantum state. An alternative method, the quantum coin (to be covered in Section 3), instead loads the function to the quantum state directly rather than its square root (something which is sometimes referred to as direct embedding (Kubo et al., 2020)). Nevertheless, both methods exploit Grover's amplification to "stretch" as much as possible the confidence interval of previous estimations. In other words, given an estimation of the desired amplitude a and a high confidence interval for it, an iterated application of the Grover operator Q allows to "zoom-in" and extend the confidence interval to the region where the sine function (4.29) is invertible. Thus, one can obtain a finer estimation of θ, and thereby of a. Further details on this "zoom-in" approach are provided in 4.1.3.1.

4.1.2.3.3. Power-law and QoPrime Amplitude Estimation Both Power-law and QoPrime algorithms have been described in (Giurgica-Tiron et al., 2020). The paper frames the asymptotic trade-off between

the quantum speedup of an amplitude estimation algorithm and its depth. More precisely, the authors relate the speedup to the total number of oracle calls $N = O\left(\frac{1}{\epsilon^{1+\beta}}\right)$, while the depth is given by the number of oracle calls that need to be performed sequentially $D = O\left(\frac{1}{\epsilon^{1-\beta}}\right)$; ϵ represent the precision (the additive error), while $0 \leq \beta \leq 1$. The extreme cases when $\beta = 0$ and $\beta = 1$ are respectively associated to the standard Quantum Amplitude Estimation algorithm and to classical Monte Carlo. Note that D is inversely related to the degree of parallelizability, and it is relevant to stress that it represents the asymptotic depth due the needed sequential calls to the oracle, without taking into account the $O(\log\log(1/\epsilon))$ depth overhead due to the eventual Quantum Fourier transform of the standard Quantum Amplitude Estimation algorithm. Roughly, Power-law and QoPrime algorithms interpolate between the quantum and classical cases playing with the trade-off between N and D, at fixed $ND = O\left(\frac{1}{\epsilon}\right)$.

The Power-law algorithm (sometimes referred to as Kerenidis-Prakash algorithm) refines the Max Likelihood algorithm of (Suzuki et al., 2020). This class of algorithms rely on a sampling schedule (m_k, N_k) where the oracle is called sequentially m_k times for N_k iterations. Eventually, the results collected according to the schedule are post-processed classically. The Power-law algorithm optimizes such sampling schedule, proposing a power-law schedule instead of an alternative exponential or linear schedule as originally suggested by (Suzuki et al., 2020). These functional forms refer to the dependence of m_k on k.

The QoPrime algorithm follows the same trade off between N and D as the Power-law algorithm, however its strategy is based on a result from number theory, that is known as Chinese Remainder Theorem. This concerns modular arithmetic and allows to combine a set of low-accuracy samplings in order to obtain a high accuracy result. The key technical point is to define a schedule based on coprime integers which, intuitively, provide independent information about the result, analogous to projections on distinct elements of an orthogonal basis in vector calculus.

4.1.3. Quantum coins

As already mentioned in Subsection 4.1.2.3, the Quantum Fourier transform requires heavy quantum circuits which are likely incompatible with near-term quantum technologies. The "quantum coin" (Aaronson & Rall, 2020; Abrams & Williams, 2004; Shimada & Hachisuka, 2020) offers a way to circumvent this issue which relies on an alternative algorithm, differing from quantum amplitude estimation in some of its basic aspects. Although belonging to the same family of Grover algorithms, the quantum coin loads the integrand function (and not its square root) into the quantum state amplitude. We stress that this apparently small modification can turn in normalization subtleties: in fact, when loading the function instead of its square root, the normalization of the state does not correspond to the normalization of the function, this being relevant in dealing with probability density distributions, as we will explicitly see in 4.1.3.2.

4.1.3.1. The original integration algorithm

The core idea is still Grover's: K out of N elements are marked ("good" states), so that one can define the Grover angle

$$\theta \equiv \arcsin\sqrt{\frac{K}{N}}\,, \tag{4.37}$$

and then exploit the "Grover amplification" to speedup the search quadratically. In quantum coins algorithm, such amplification is implemented as a zoom-in operation of the confidence level (Abrams & Williams, 2004; Shimada & Hachisuka, 2020). To appreciate this, we need to delve into some details.

Let us consider an initial state

$$|\Psi_0\rangle = |0\rangle_n|0\rangle\,, \tag{4.38}$$

where the second register is an auxiliary qubit, while the first register is composed of n qubits encoding the (discretized) domain of integration.[10] We first apply a Walsh-Hadamard transform on the second register

$$|\Psi_1\rangle = \left(H^{\otimes n}\otimes \mathbb{1}\right)|\Psi_0\rangle = \frac{1}{2^{\frac{n}{2}}}\sum_{x=0}^{2^n-1}|x\rangle_n|0\rangle\,. \tag{4.39}$$

[10] Some comments on the discretization and labelling of the integration domain were given at the beginning of 4.1.2.1.

Here our purpose is estimating the integral of a given function $f(x)$. More precisely, we want to compute (4.5), i.e.

$$\sum_x f(x) \,, \tag{4.40}$$

for x running on the discretization of the domain implemented by the first quantum register in (4.38). Assume that we are able to define (and implement) a quantum operator which "loads" the function f into a quantum state. In other words, we suppose to have an "oracle" acting as follows:

$$\begin{aligned} |\Psi_2\rangle &= O_f|\Psi_1\rangle \\ &= \frac{1}{2^{\frac{n}{2}}}\sum_x \left[\sqrt{1-f(x)^2}\,|x\rangle_n|0\rangle + f(x)|x\rangle_n|1\rangle\right] \,. \end{aligned} \tag{4.41}$$

We stress once more that we are loading the function, and not its square root, into the amplitudes featuring the auxiliary qubit on 1. A further application of the Walsh-Hadamard operator "collects" the relevant contribution coherently into the $|0\rangle_n|0\rangle$ component. Explicitly, we have

$$\begin{aligned} |\Psi_3\rangle &= \left(H^{\otimes n}\otimes \mathbb{1}\right)|\Psi_2\rangle \\ &= \frac{1}{2^n}\sum_x \left[\sqrt{1-f(x)^2}|0\rangle_n|0\rangle + f(x)|x\rangle_n|1\rangle\right] + \dots \,, \\ &= \left(\frac{1}{2^n}\sum_x \sqrt{1-f(x)^2}\right)|0\rangle_n|0\rangle + \left(\frac{1}{2^n}\sum_x f(x)\right)|0\rangle_n|1\rangle + \dots \end{aligned} \tag{4.42}$$

where the final ellipsis ... correspond to other quantum states we are not directly interested in. In summary, the Walsh-Hadamard operation applied in (4.42) leads to a quantum state $|\Psi_3\rangle$ whose component along $|0\rangle_n|1\rangle$ is given by the integral we want to compute.

The implementation up to (4.42) is a quantum version of tossing a classical coin (being heads and tails associated to measuring either $|0\rangle_n|0\rangle$ on the one side or any other outcome on the other). So far, no quantum advantage is implied by the quantum implementation. In order to improve it, one can rely again on Grover's amplification. More specifically, we first perform an initial estimation with the unamplified Qcoin, then we rely on Grover amplification to progressively refine the result, as described in the subsections below.

First estimation

We first observe that the Walsh-Hadamard operator $H^{\otimes n}$, when acting on the initial state, "loads" the uniform probability distribution as follows

$$|\Psi\rangle = (H^{\otimes n}\otimes \mathbb{1})|0\rangle_n|0\rangle = \frac{1}{2^{\frac{n}{2}}}\sum_x |x\rangle_n|0\rangle \,, \tag{4.43}$$

where the second label in the quantum state refers to an auxiliary qubit, used later in order to load the function $f(x)$; n is the number of qubits in the physical register. According to (4.41), we recall that we also assume to have an oracle loading the function $f(x)$, acting on the individual state $|x\rangle_n|0\rangle$ as follows

$$O_f|x\rangle_n|0\rangle = f(x)|x\rangle_n|1\rangle + \left[1-f(x)^2\right]^{\frac{1}{2}}|x\rangle_n|0\rangle \,. \tag{4.44}$$

From the definition of the state $|\Psi\rangle$ in (4.43) and the action of the oracle given in (4.44), we have

$$O_f|\Psi\rangle = \frac{1}{2^{\frac{n}{2}}}\sum_x f(x)|x\rangle_n|1\rangle + \frac{1}{2^{\frac{n}{2}}}\sum_x \left[1-f(x)^2\right]^{\frac{1}{2}}|x\rangle_n|0\rangle \,. \tag{4.45}$$

Applying a Walsh-Hadamard operator on the quantum register (leaving the auxiliary qubit untouched), we get

$$|\Phi\rangle \equiv \left(H^{\otimes n}\otimes \mathbb{1}\right)O_f|\Psi\rangle = \frac{1}{2^n}\sum_x f(x)|0\rangle_n|1\rangle + \dots \,. \tag{4.46}$$

The Walsh-Hadamard transform "collected" the value that we want to compute into the amplitude of the component $|0\rangle_n|1\rangle$. The dots refer to other components on which we do not need to focus the attention, their amplitudes are analogous sums with all the possible combinations of minus and plus signs.

Using

$$(|0\rangle|1\rangle)^\dagger = \langle 1|\langle 0| \, , \tag{4.47}$$

where we have dropped the subindex n, we have

$$\mathbb{E}\left(f(x)\right) = \sum_x f(x) = 2^n \, \langle 1|\langle 0| \, |\Phi\rangle \, , \tag{4.48}$$

so from the probability P_{01} of measuring $|0\rangle_n|1\rangle$ in $|\Phi\rangle$, we can derive the desired result

$$|\mathbb{E}\left(f(x)\right)| = 2^n \sqrt{P_{01}} \, . \tag{4.49}$$

Note that we are getting the absolute value of the expected value, losing information about its sign. This is however not a problem when (as it happens for option pricing tasks) the function to be integrated is non-negative across its whole domain.[11]

In order to get a first empirical estimation $\tilde{\mu}$ of (4.49) we run the previous quantum computation m times and assign a confidence interval relying on standard statistical tools, *e.g.* Chebyshev's inequality (see 4.1.3.1). Thus, the result of the first estimation is given by a confidence interval centered about $\tilde{\mu}$, namely

$$[\tilde{\mu} - \epsilon, \tilde{\mu} + \epsilon]. \tag{4.50}$$

Subsequent iterations

The strategy of the QCoin algorithm is based on two operations:

- A shift of the function $f(x)$ by the lower bound of the (estimated) confidence interval (4.50), so to get a function $\bar{f}(x)$ for which we expect a positive mean value, which constitutes a refinement of our previous estimation.
- An enhancement of the probability P_{01} of getting the $|0\rangle|1\rangle$ state. This is the key step, related to amplitude amplification. The idea is as follows: by means of Grover amplification we obtain a state where the probability $\hat{P}_{01}$ of getting the $|0\rangle|1\rangle$ is

  $$\hat{P}_{01} = \alpha P_{01} \, , \tag{4.51}$$

  for $\alpha > 1$. An astimation of $\hat{P}_{01}$ with precision ϵ corresponds to an estimation of P_{01} with increased precision $\frac{\epsilon}{\alpha}$. See Subsection 4.1.3.1 for details.

To recapitulate, one could apply standard quantum amplitude estimation to get (4.49), but the purpose is here to avoid all the coherent manipulations needed to implement the inverse Quantum Fourier transform, namely we aim to obtain the same result with shallower circuits.

Let us first observe that, given a constant c, we have

$$\mathbb{E}\left(f(x) - c\right) = \mathbb{E}\left(f(x)\right) - c \, , \tag{4.52}$$

where we relied on the normalization of the probability distribution.[12] In particular, we want to shift the original function by the (empirical) lower bound of the confidence interval (4.50), namely

$$\bar{f}(x) \equiv f(x) - (\tilde{\mu} - \epsilon) \, . \tag{4.54}$$

[11] Also in more general cases, one can exploit the finiteness of the function $f(x)$ and the linearity of the integral to shift and rescale $f(x)$ so to pose the problem as the integral of a positive function.

[12]
$$\mathbb{E}\left(f(x) - c\right) = \sum_x p(x)\left[f(x) - c\right] = \left[\sum_x p(x) f(x)\right] - c = \mathbb{E}\left(f(x)\right) - c \tag{4.53}$$

We assume to know how to shift the function $f(x)$ vertically, namely, being able to build the orcale

$$O_{\bar{f}}|x\rangle_n|0\rangle = \bar{f}(x)|x\rangle_n|1\rangle + \left[1-\bar{f}(x)^2\right]^{\frac{1}{2}}|x\rangle_n|0\rangle \ . \tag{4.55}$$

Thus, we can proceed as before, namely we can evaluate the expected value of $\bar{f}(x)$ as follows:

$$\mathbb{E}\left(\bar{f}(x)\right) = 2^n \langle 1|\langle 0|\,|\bar{\Phi}\rangle \ , \tag{4.56}$$

where

$$|\bar{\Phi}\rangle = \left(H^{\otimes n}\otimes \mathbb{1}\right) O_{\bar{f}}|\Psi\rangle \ , \tag{4.57}$$

in analogy with (4.49). To avoid clutter, in (4.56) and in the following formulæ, we drop the subindex n for the physical register (*i.e.* the first of the two kets or the second of the two bras) and accordingly we denote the Hermitian conjugation of tensor products of generic kets $|a\rangle$ and $|b\rangle$ as

$$(|a\rangle|b\rangle)^\dagger = \langle b|\langle a| \ . \tag{4.58}$$

For the moment we have not considered any amplification, we have just shifted the function by the lower bound of the confidence interval, $\tilde{\mu}-\epsilon$, so to expect positive refinements. To pursue this direction, we define the Grover amplificator as usual

$$Q_{\bar{f}} = O_{\bar{f}}\, R_{00}\, O_{\bar{f}}^{-1}\, R_{01} \ , \tag{4.59}$$

where R_{00} and R_{01} are the reflection operators given by

$$R_{00} = \mathbb{1} - 2|0\rangle|0\rangle\ \langle 0|\langle 0| \ , \tag{4.60}$$

$$R_{01} = \mathbb{1} - 2|0\rangle|1\rangle\ \langle 1|\langle 0| \ . \tag{4.61}$$

Defining

$$\sin(\theta) \equiv \langle 1|\langle 0|\,|\bar{\Phi}\rangle \ , \tag{4.62}$$

we have

$$\langle 1|\langle 0|Q^k|\bar{\Phi}\rangle = \sin\left((2k+1)\theta\right) \ . \tag{4.63}$$

The corresponding amplification factor α for the probability of measuring $|0\rangle|1\rangle$ is

$$\alpha = \frac{\sin\left((2k+1)\theta\right)^2}{\sin(\theta)^2} \ . \tag{4.64}$$

In order to maximize α we need to chose k so that

$$(2k+1) \lesssim \frac{\pi}{2} \ . \tag{4.65}$$

Thus, we will collect all the possible measurment outcomes of the quantum circuit, and -having amplified the probability of getting $|0\rangle|1\rangle$ in a controlled way- we can estimate θ from the measurment statistics. The acceleration being provided by the amplification (4.64).

To recapitulate, we are here describing an implementation of the approach proposed in (Abrams & Williams, 2004) and further developped in (Shimada & Hachisuka, 2020). Answering to the same question raised in (Aaronson & Rall, 2020), the algorithm above represents a simplified method to perform amplitude estimation.

Confidence interval

Chebyshev's inequality bounds the probability for the empirical mean value of deviating from the true mean value by more than ϵ,

$$\mathrm{Pr}\left(\left|\frac{\sum_i X_i}{n} - \mu\right| \geq \epsilon\right) \leq \frac{\sigma^2}{n\epsilon^2} \ , \tag{4.66}$$

where μ is the true mean value, σ^2 is the true variance, n is the number of sampling and X_i (the samples) are i.i.d. random variables. Consider them to be Bernoullian variables with $\mathrm{Pr}(X_i = 1) = p$. Thus the

variance is

$$\sigma^2 = p(1-p) \leq \frac{1}{4} \,, \tag{4.67}$$

bounded by the case where $p = \frac{1}{2}$, *i.e.* the fair coin. Using inequality (4.67), we can attribute confidence intervals according to the following weakened version of Chebyshev's inequality

$$\text{Pr}\left(\left| \frac{\sum_i X_i}{n} - \mu \right| \geq \epsilon \right) \leq \frac{1}{4n\epsilon^2} \,. \tag{4.68}$$

This will be a conservative method, liable to possible refinements. Suppose we fix an error ϵ and a confidence level w (*i.e.* the probability of falling within $\pm\epsilon$ of the correct result), then the required number of trials goes as

$$n \sim \frac{1}{4(1-w)\epsilon^2} \tag{4.69}$$

Recalling that we have

$$\hat{P}_{01} = \alpha \, P_{01} \,, \tag{4.70}$$

with $\alpha > 1$, the confidence interval is shrunk according to

$$\left[\hat{P}_{01} - \epsilon, \hat{P}_{01} + \epsilon \right] \;\; \rightarrow \;\; \left[P_{01} - \frac{\epsilon}{\alpha}, P_{01} + \frac{\epsilon}{\alpha} \right] \,. \tag{4.71}$$

A couple of additional comments:

- The algorithm described above relies on a similar strategy as the quantum iterative amplitude estimation (Grinko et al., 2020).
- The analysis done on the basis of Chebyshev's inequality can be possibly refined relying on Chernoff bound (Cover, 2005; Hoeffding, 1963) or Clopper-Pearson bound (Clopper & Pearson, 1934; Scholz, 2008)

4.1.3.2. Generalization to a non-uniform distribution

In order to apply the Qcoin method to an option pricing model, it needs to be generalized to the case of a non-uniform distribution. Said otherwise, we need to compute the expectation of a payoff function $f(x)$ with respect to a generic distribution $p(x)$,

$$\mathbb{E}\left(f(x)\right) = \sum_x p(x) f(x) \,. \tag{4.72}$$

The reason why such generalization is necessary, and we cannot just apply the Qcoin to the entire integrand function $p(x)f(x)$, derives from the fact that it is difficult to define an oracle for the product function $p(x)f(x)$. This is currently work in progress for the WP5.

4.1.4. An option pricing example adopting amplitude amplification and estimation

As presented Equation in (3.8), the option pricing problem can be formulated as the computation of the expected value of a payoff function with respect to a probability distribution.[13] The simplest use-cases are provided by vanilla European options (see Section 3.1.1.1, for details) and are considered in the literature as a first benchmark, or better proof-of-concept, for quantum implementation, see for example (Kaneko et al., 2020b; Ramos-Calderer et al., 2020; Rebentrost et al., 2018; Stamatopoulos et al., 2020).

We follow (Stamatopoulos et al., 2020). We need to define an operator $\mathcal{A}$ such that a in (4.17) corresponds to the expected value of the payoff function $f(x)$,

$$a = \sum_{x=0}^{2^n-1} f(x)p(x) = \mathbb{E}\left(f(x)\right) \,, \tag{4.73}$$

[13] This is true also in path dependent cases, but one must refer to the probability distribution defined on the path space.

and compute a by means of amplitude estimation (Brassard et al., 2000).

Vanilla options have piece-wise linear payoff functions $f(x)$. By means of controlled Y rotations, one can efficiently load the linear function $f(x)$ to a quantum state. Suppose that $f(x) = f_1 x + f_0$. Consider the operator

$$O_f \equiv R_y(f_0) \prod_{i=1}^{n} cR_y\left(|x_{(i)}\rangle, |\text{aux}\rangle; 2^i\right) , \tag{4.74}$$

where $cR(|A\rangle, |B\rangle, \varphi)$ implements a rotation by an angle φ on the qubit $|B\rangle$ controlled by the qubit $|A\rangle$; with $x_{(i)}$ we have denoted the i-th digit in the binary representation of x. Thus, we have

$$O_f|x\rangle|0\rangle = |x\rangle \left[\cos(f(x))|0\rangle + \sin(f(x))|1\rangle\right] . \tag{4.75}$$

Note that we have loaded the function into the argument of the sine in the amplitudes of the "marked" qubits, so we need to linearize it. To this purpose, consider the following deformed function

$$\tilde{f}(x) \equiv c\left[2\frac{f(x) - f_{\text{min}}}{f_{\text{max}} - f_{\text{min}}} - 1\right] + \frac{\pi}{4} , \tag{4.76}$$

where c is a rescaling parameter that we will want to keep small,

$$c\left|2\frac{f(x) - f_{\text{min}}}{f_{\text{max}} - f_{\text{min}}} - 1\right|_{\text{max}} \ll \frac{\pi}{4} . \tag{4.77}$$

The deformed function $\tilde{f}$ takes values in the interval $[\frac{\pi}{4} - c, \frac{\pi}{4} + c]$. We build the operator $O_{\tilde{f}}$ in analogy to (4.75).

Consider an arbitrary probability distribution and load it to a quantum state,

$$|\psi_0\rangle = \sum_x \sqrt{p(x)}\,|x\rangle|0\rangle . \tag{4.78}$$

By applying $O_{\tilde{f}}$ we get

$$O_{\tilde{f}}|\psi_0\rangle = \sum_{x=0}^{2^n-1} \sqrt{p(x)}\,|x\rangle\left[\cos\left(\tilde{f}(x)\right)|0\rangle + \sin\left(\tilde{f}(x)\right)|1\rangle\right] . \tag{4.79}$$

The probability of measuring a good state is given by

$$P_1 = \sum_{x=0}^{2^n-1} p(x)\sin^2\left(\tilde{f}(x)\right) , \tag{4.80}$$

which can be approximated as follows

$$\begin{aligned} P_1 &\approx \sum_{x=0}^{2^n-1} p(x)\left\{c\left[2\frac{f(x) - f_{\text{min}}}{f_{\text{max}} - f_{\text{min}}} - 1\right] + \frac{1}{2}\right\} \\ &= 2c\frac{\mathbb{E}[f(x)] - f_{\text{min}}}{f_{\text{max}} - f_{\text{min}}} - c + \frac{1}{2} \end{aligned} \tag{4.81}$$

when (4.77) holds.

To implement a piece-wise linear function, we need extra auxiliary qubits to label the intervals where the function is linear. In the case of two regions we need just one extra auxiliary qubit. This is the case for the payoff of a vanilla European call option

$$f(x) = \max(x - K, 0) \tag{4.82}$$

The splitting into two regions can be realized by means of a comparator (see (Cuccaro et al., 2004)) we get

$$|\psi_1\rangle = \sum_{x<K} \sqrt{p(x)}\ |x\rangle|0\rangle + \sum_{x\geq K} \sqrt{p(x)}\ |x\rangle|1\rangle\ . \tag{4.83}$$

Then we upload the linear pieces as shown below. Specifically we upload the function

$$g(x) = h_0 + [g_0 + g_1(x-K)]\ \Theta\left(x-K\right)\ , \tag{4.84}$$

whose parameters h_0, g_0 and g_1 are to be specified below, while $\Theta\left(x-K\right)$ is vanishing for $K \geq x$ and equal to 1 for $K < x$ (Θ is a Heaviside function). We have

$$\begin{aligned} O_g|\psi_1\rangle|0\rangle = &\sum_{x<K} \sqrt{p(x)}\ |x\rangle|0\rangle\ [\cos(h_0)|0\rangle + \sin(h_0)|1\rangle] \\ &+ \sum_{x\geq K} \sqrt{p(x)}\ |x\rangle|1\rangle\ [\cos(h_0+g_0+g_1x)|0\rangle + \sin(h_0+g_0+g_1x)|1\rangle]\ . \end{aligned} \tag{4.85}$$

The probability of getting 1 for the last qubit by measuring the state $O_g|\psi_1\rangle|0\rangle$ is given by

$$P_1 = \sum_{x<K} p(x)\sin^2(h_0) + \sum_{x\geq K} p(x)\sin^2(h_0+g_0+g_1x)\ . \tag{4.86}$$

Comparing the specific payoff given in (4.82) with the case for generic $f(x)$, (4.76), we have

$$2c\frac{x-K}{x_{\text{max}}-K} - c + \frac{1}{2} = h_0 + g_0 + g_1x\ , \tag{4.87}$$

and

$$g_1 = \frac{2c}{x_{\text{max}}-K}\ , \tag{4.88}$$

$$g_0 = -2c\,\frac{K}{x_{\text{max}}-K}\ , \tag{4.89}$$

$$h_0 = \frac{\pi}{4} - c\ . \tag{4.90}$$

We refer to (Stamatopoulos et al., 2020) for further details.

4.1.5. Applications to risk analysis

As described in Section 3.2, some of the tools used for options pricing can be leveraged for financial risk analysis.

In particular, both in the pricing and in the risk assessment arena, Monte Carlo methods (see Sections 3.1.3.1 and 3.2.4.1) play a predominant role. Nevertheless, risk analysis problems require a precise estimation of the tail of a distribution, which constitute a *more* demanding regime in terms of Monte Carlo samplings. On top of that, the problem usually is highly dimensional, as portfolios of derivatives are considered.

Classical mitigation strategies resort to importance sampling, but, also after such mitigation, the problem remains usually very heavy in terms of needed resources. Because of this state of affairs, a possible improvement in efficiency due to a quantum algorithm results particularly interesting in the field of financial risk assessment.

A generally important technical ingredient both in option pricing and in risk assessment is given by the computation of expected values above (or below) a pre-specified bound. For instance, vanilla European options with a specified strike price K, feature a payoff function which "activates" above/below it, depending whether we are considering a call or put option. As already seen in Section 4.1.4, such a discontinuous activation can be implemented by means of a *quantum comparator circuit*, see (Cuccaro et al., 2004) for details.

In risk assessment one can be interested in estimating the probability of experiencing a future loss exceeding a pre-determined value, according to, for example, the definition of the VaR and th $CVaR$ risk measures as in Section 3.2.2. Such a question can again be addressed by means of a comparator. More often, one is interested in fixing a (high) confidence level α and ask which maximal loss corresponds to it. This question can be addressed combining a comparator with a binary search algorithm (Egger, Gutiérrez, et al., 2020; Egger & Woerner, 2019).

As a final remark on quantum-enhanced Monte Carlo techniques, we refer to (An et al., 2020) for the quantum generalization of the classical multi-level Monte Carlo strategy (Giles, 2015). This latter consists in an optimized sampling which favours the collection of many low-precision/low-cost samples and entails the collection of just a few high-precision/high-cost samples. Such multi-level strategy is encoded in a telescopic sum where each term represents a Monte Carlo sub-problem. The idea of (An et al., 2020) is to apply a quantum circuit to solve each Monte Carlo sub-problem.

4.2. Other methods with quantum computing

4.2.1. Quantum algorithms to solve the Black-Scholes partial differential equation

Following (Gonzalez-Conde et al., 2021), the quantum approaches rely on the observation that the financial PDEs can be mapped into the propagation according to an appropriate Hamiltonian operator.

For this purpose, the first step is to consider appropriate changes of variable and/or unknown in the Black-Scholes equation (3.13). Note that this is the usual way to reduce this equation to a PDE with constant coefficients or to a heat equation. Also, this technique is used in European vanilla options to obtain the Black-Scholes formula.

First, by using the change of variable $S = e^x$, equation (3.13) becomes

$$\frac{\partial V}{\partial t} + \left(\mu - \frac{\sigma^2}{2}\right)\frac{\partial V}{\partial x} + \frac{\sigma^2}{2}\frac{\partial^2 V}{\partial x^2} - \mu V = 0 \,, \tag{4.91}$$

which can be written as

$$\frac{\partial V}{\partial t} = -i\hat{H}_{\text{BS}}\, V \,, \tag{4.92}$$

where

$$\hat{H}_{\text{BS}} = i\frac{\sigma^2}{2}\hat{p}^2 - \left(\frac{\sigma^2}{2} - \mu\right)\hat{p} + i\mu\mathbb{1} \,, \tag{4.93}$$

and we have defined the momentum operator

$$\hat{p} = -i\frac{\partial}{\partial x} \,. \tag{4.94}$$

Note that (4.92) is a Schrödinger-like equation. However, it is important to stress that the Hamiltonian $\hat{H}_{\text{BS}}$ defined in (4.93) is *not* Hermitian. Therefore, the associated evolution operator

$$\hat{U}(t, t_0) = e^{-i\hat{H}_{\text{BS}}(t-t_0)} \,, \tag{4.95}$$

is not unitary. For implementing the evolution operator (4.95) into a quantum circuit, *i.e.* through unitary operations, one can consider an enlarged system.[14] In (Gonzalez-Conde et al., 2021), $\hat{U}(t, t_0)$ is embedded into a doubled unitary operator where the doubling is implemented at the price of adding an auxiliary qubit with respect to which one needs to post-select.

An interesting alternative is followed in (Fontanela et al., 2021). Instead of doubling the systems, one can perform an additional change of variable $\tau = \sigma^2(T - t)$ and consider a new unknown $v(x, \tau) = \exp(-ax - b\tau)V(t, s)$, with appropriate constant values of a and b (see (Vázquez, 2010) for details) so that (4.91) maps into the heat equation

[14] This is a standard technique followed in quantum mechanics (and quantum field theory) when dealing with a subsystem in contact with an external environment. For instance, to consider a quantum system in contact with a thermal bath or when one want to consider dissipation.

$$\frac{\partial v}{\partial \tau} = \frac{1}{2}\frac{\partial^2 v}{\partial x^2}\,. \tag{4.96}$$

Next, using the Wick rotation $\tilde{\tau} = -i\tau$, which maps real time to imaginary time, the heat equation (4.96) turns into a Schrödinger-like one, namely

$$\frac{\partial v}{\partial \tilde{\tau}} = -\hat{H}_{\mathsf{HE}}\, v\,, \tag{4.97}$$

with

$$\hat{H}_{\mathsf{HE}} = -\frac{i}{2}\hat{q}^2\,, \quad \hat{q} = -i\frac{\partial}{\partial x}. \tag{4.98}$$

This leads to a purely anti-Hermitian Hamiltonian operator. Said otherwise, (4.97) encodes a Hamiltonian evolution along imaginary time, that is, the Wick rotated version of a normal real-time propagation. Intuitively, imaginary-time propagation transforms oscillations into dampings, so that (4.97) can be associated to a non-unitary (read dissipative) cooling evolution. These observations are relevant in practice, especially because they allow to connect to an area where similar problems have been thoroughly investigated, namely that of finding the ground state of quantum system. This is a central problem in condensed matter physics and in chemistry, which also connects to optimization.

We briefly revise the two approaches just described above separately, commenting the associated literature.

4.2.1.1. Propagating with $\hat{H}_{\mathsf{BS}}$

In (Gonzalez-Conde et al., 2021) they consider the Hamiltonian (4.93) and exploit the fact that it is diagonal in momentum space. Thus, by means of a quantum Fourier transform, and its inverse, they are able to work with a diagonal propagator, which admits an efficient decomposition in the Cartan basis. They study the possibility of truncating the Hamiltonian retaining just a polynomial number of interactions and, on this basis, they claim an exponential speedup in the Hamiltonian propagation subroutine. Nevertheless, an overall exponential speedup for the entire pipeline would require efficient loading of the uncertainty model and of the pay-off function. These issues remain as open problems. A schematic depiction of the algorithm is given in Fig.5.

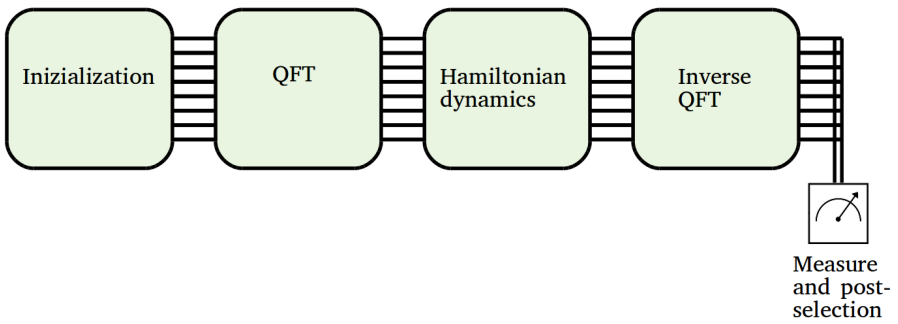


***Figure 5**: Schematic pipeline of the quantum algorithm described in (Gonzalez-Conde et al., 2021). The acronym QFT stands for Quantum Fourier Transform.*

There are two relevant drawbacks of the method, one theoretical and the other one practical. The former is related to the fact that the diagonalization in momentum space of the Hamiltonian is a "delicate" condition, spoiled when considering interest rates or volatilities which depend on the underlying asset value. That is, it is difficult to generalize the method to models which are not just Black-Scholes ones. On the practical level, the quantum Fourier transform (and its inverse too) are gate-wise demanding and easily incompatible with actual implementation in NISQ devices.

Two technical aspects of the analysis in (Gonzalez-Conde et al., 2021) are worth stressing. The first is that they double the spatial direction on which they solve the Black-Scholes equation, so that they mitigate the possible spurious effects arising from the borders. The doubling is carried out by the addition of an extra auxiliary qubit. The second technical aspect is that the Hamiltonian that they consider can be expressed

using only the diagonal Pauli matrices σ_0 and σ_z (*i.e.* the generators of the $SU(2)$ Cartan subalgebra) and there is no need to take a Trotter approximation for non-commuting terms.

4.2.1.2. Imaginary-time propagation with $\hat{H}_{\text{HE}}$

A quantum algorithm for imaginary-time propagation has been developed in the field of quantum chemistry (McArdle et al., 2019).[15] It assumes to deal with a Hamiltonian

$$\hat{H} = \sum_i \lambda_i \hat{h}_i \ , \tag{4.99}$$

given by a polynomial number of terms where the coefficients λ_i are real and the operators $\hat{h}_i$ are observables which admit an expression in terms of tensor products of Pauli operators. In (Fontanela et al., 2021) such assumption is imported into the financial application domain, although it is not discussed in detail.

The imaginary-time evolution of the quantum state needs special care due to its lack of unitarity. In (McArdle et al., 2019) they address this aspect by means of a suitable normalization factor.

The quantum state is then approximated with a parametric circuit, like in the variational quantum eigensolver approach (VQE). Nevertheless, as opposed to this latter, the parameters Θ of the circuit are not optimized. In fact, the idea of the imaginary-time propagation method is essentially to trade an optimization with a cooling (or annealing) driven by a Hamiltonian evolution. Specifically, if the initial state overlaps with the ground state, and if the circuit ansatz is able to represent the ground state, then the imaginary-time evolution leads the system to eventually land on the ground state. The approach is attractive because it avoids the circuit optimization, whose hardness and scaling properties are difficult to assess. Nevertheless, some difficulties are translated into the choice of the ansatz and to its capability of expressing and reaching the ground state efficiently.

More technically, the imaginary-time method entails considering a McLachlan variational principle

$$\delta \left|\left| \left(\frac{\partial}{\partial \tau} + \hat{H} + E \right) |\psi(\Theta(\tau))\rangle \right|\right| = 0 \ , \tag{4.100}$$

where we remind the reader that Θ are the parameters of the circuit ansatz. The coefficient E is related to the above-mentioned normalization issue, see (McArdle et al., 2019) for details.

The evolution of the parameters is derived from the linear system of differential equations associated to the variational principle (4.100), namely

$$\sum_j A_{ij} \dot{\theta}_j = b_i \ , \tag{4.101}$$

where $\dot{\theta}_j = \partial \theta_j / \partial \tau$ and

$$A_{ij} = \text{Re} \left[\frac{\partial \langle \psi |}{\partial \theta_i} \frac{|\partial \psi\rangle}{\partial \theta_j} \right] \ , \tag{4.102}$$

$$b_i = -\text{Re} \left[\sum_a \lambda_a \frac{\partial \langle \psi |}{\partial \theta_i} \hat{h}_a |\psi\rangle \right] \ . \tag{4.103}$$

The matrix A_{ij} and the vector b_i are claimed to be efficiently computable with a quantum subroutine embedded in the overall, hybrid quantum/classical algorithm.

Also this approach presents some drawbacks. First, the transformation to a pure heat equation, (4.96), occurs for the Black-Scholes model but is expected not to hold when generalizing it. Secondly, the evolution of the parameter is governed by (4.101) which is solved with classical computing techniques. Its efficiency and scaling properties have not been assessed thoroughly.

It is interesting to explore the possibility of solving (4.101) still within the quantum circuit, possibly implementing the Harrow-Hassidim-Lloyd (HHL) algorithm (Harrow et al., 2009) or its refinements (see for

[15] For related discussions we refer to (Tan, 2020) and (Hirofumi Nishi & Matsushita, 2020).

example (Carrera Vazquez et al., 2020)). For further discussions about solving partial differential equations in quantum computers we refer to (García-Molina et al., 2021) and (Kyriienko et al., 2021).

4.2.2. (Quantum) Machine Learning

As for many other scientific disciplines, in the last decade machine learning techniques have been intensively applied to diverse problems in quantitative finance. Regression-based pricing methods, PDE resolution and optimal stochastic control problems are prominent examples. For that reason, the recent advances in the so-called Quantum Machine Learning (QML) can have a great impact when employed on pricing derivatives and risk management or other relevant tasks for the financial industry. The QML explores how to devise and implement quantum algorithms that outperform classical computers on machine learning tasks (Biamonte et al., 2017). Multiple machine learning classical components have been recently adapted to quantum systems, opening this way a full range of novel applications. Although these new QML algorithms have not been widely employed, so far, for financial applications (to the best of the authors' knowledge), they deserve to be considered in the near future.

In the following, we summarised the most promising developments in the QML area, which could be potentially applied to problems appearing in the financial sector, particularly the ones described in Chapter 3. Note however that most quantum machine algorithms working with classical data assume the availability of a Quantum Random Access Memory (QRAM), which are not expected to be physically realizable in the near future (Bouland et al., 2020). For a discussion about the relation among quantum machine learning and kernel methods we refer to (Schuld, 2021).

4.2.2.1. Quantum Principal Component Analysis

In Section 3.2.4.3 we have briefly described the usage of the PCA algorithm in finance. The basic objective behind PCA is to calculate the eigenvalues of the covariance matrix $\Sigma \in \mathbb{R}^N \times \mathbb{R}^N$, which is Hermitian positive semi-definite. For this purpose, one possibility is to use the Quantum Phase Estimation (QPE) algorithm to obtain the eigenvalues. However, two problems arise in this case: an initial state has to be initialized which includes all the eigenvectors of the covariance matrix and the covariance matrix must be decomposed on a summation of Pauli strings (which needs N^2 classical operations). Taking into account that Σ is usually a dense matrix , there is no guarantee of obtaining a radical speed-up as could be achieved in other algorithms like HHL, for example.

Another possibility is to work with the covariance matrix Σ as a density matrix (usually represented as ρ), and make a Quantum Principal Component Analysis (QPCA). The initial proposal of this approach is due to (Lloyd et al., 2014), who describe an algorithm to obtain the exponential of a density matrix using several copies (C) of it. One example of this technique is the QPCA, which was implemented by (Abhijith et al., 2020) in the case $N = 2$, with $C = 2$. Having enough resources, it theoretically runs exponentially faster. Other refined versions have been proposed later (He et al., 2021; Lin et al., 2019).

Recently, (Xin et al., 2021) suggested to calculate the eigenvalues and eigenvectors of Σ by using a variational algorithm, by using the density matrix expansion

$$\rho = \sum_{j=0}^{N-1} \lambda_j |\psi_j\rangle\langle\psi_j|, \tag{4.104}$$

where $\{\psi_j\}$ represents an orthogonal basis. Thus, it is possible to find a unitary transformation such that:

$$\rho_f = U_g(\Theta)\rho U(\Theta)_g^\dagger = \sum_{j=0}^{N-1} \lambda_j |j\rangle\langle j|, \tag{4.105}$$

where $\{|j\rangle\}$ is the usual computational basis and $\Theta = \{\theta_i\}$ is the set of angles that define the operator U. This unitary operator is searched for by optimizing the parameters Θ using variational hybrid algorithms. Once $U(\Theta)_g$ is known, the eigenvalues can be computed directly by measuring probabilities on the computational basis. However, it is not clear yet if $U(\Theta)$ is an efficient operator for the general case.

The workflow to use these density-matrix-based QPCA algorithms should include several steps (Abhijith et al., 2020):

- Convert Σ in a density matrix ρ. For this purpose, two characteristics of the density matrix must be fulfilled: it must be Hermitian positive semi-definite and its trace must be equal to 1 ($tr(\rho) = 1$). As the covariance matrix Σ is Hermitian and positive semi-definite, only a division by its trace is needed ($\rho = \Sigma/tr(\Sigma)$) to convert it into a density matrix. This step consumes N^2 classical operations.
- As Quantum Computers can only work with pure states, in general, ρ must be purified because it can represent mixed states. This means that to load ρ into a quantum circuit, $n = 2\log(N)$ qubits and additional classical preprocessing are required.
- This purified state must be loaded in the states, which could need a large number of gates (see 4.3.1 section for a discussion about the loading problem).

In general, these facts limit the scalability of such QPCA to $O(N^2)$ operations. However, it could exhibit still better performance than the general classical complexity of $O(N^3)$.

4.2.2.2. Regression

Classical and more advanced (neural networks-based) regression methodologies are greatly appreciated in derivatives pricing problems, in particular for options with early exercise, like American options (see 3.1.1.2), when they are priced by Monte Carlo methods. The plain regression algorithms rely on solving linear systems, a task of enormous interest in quantum computation. The HHL algorithm proposed in (Harrow et al., 2009) is a first relevant representative one for solving linear systems, allowing to diagonalize some special matrices with exponential speedup. Then, the HHL algorithm was employed to perform a regression on a quantum computer in (Wiebe et al., 2012). Some related works in this field are (Schuld et al., 2016; Wang, 2017). A more involved approach is presented in (Reddy & Bhattacherjee, 2021), where the authors tested several existing quantum (machine learning) regression algorithms tailored to a specific problem in chemistry, like *Quantum radial basis function neural network* (Shao, 2020) or *Quantum Neural Networks* (Beer et al., 2020) (and the references therein). Other quantum-based techniques like *Quantum Kernel Estimation* (Egger, Gambella, et al., 2020) and QML with Gaussian processes (Zhao et al., 2019) have been recently proposed.

4.2.2.3. (Quantum) Neural Networks and deep learning

The drastic increase of the computational power has enabled the use of deep ANN for general purpose applications, giving rise to the so-called *deep learning* techniques. Within this framework, we can identify supervised learning, unsupervised learning, reinforcement learning, convolutional networks, etc. As it is the case for many other disciplines, computational finance greatly benefits from this new computational paradigm, with successful developments on numerical solutions for PDEs or Backward Stochastic Differential Equations (BSDE), inverse option pricing problems, counterparty credit risk computations, etc. In Section 3.1.3.3, we have presented a PDE-based problem formulation suitable to be solved by unsupervised ANN approach.

In the context of deep learning and ANN, the quantum advantage can be exploited from different points of view. The first and most obvious contribution is achieved by improving the training procedure employing quantum computers. The use of, for example, quantum algorithms can have a positive impact on increasing the training computational efficiency and/or avoiding possible undesirable malfunctioning (local minima) with respect to the classical alternatives (stochastic gradient descent, backpropagation, etc.). The advances on training the so-called *Boltzmann machines* (particularly the Restricted Boltzmann Machine (RBM)) are specially relevant. Some representative works in this area of interest are (Adachi & Henderson, 2015; Alcazar et al., 2020; Job & Adachi, 2020; Vinci et al., 2020).

Another research line consists of the algorithms based on a fully Quantum Neural Network (QNN). In the last few years, many works on QNN advanced training have been proposed, among which we highlight (Beer et al., 2020; Coyle et al., 2021). One of the main applications of QNNs is the function (or distribution) loading. For its great importance in the financial problems, we devote a specific section to this particular aspect (see Section 4.3.1).

4.3. Discussion

4.3.1. The loading problem

Loading the necessary state into a quantum register in order to initialize the algorithm constitutes in many cases the main bottleneck. For example, in Grover Amplification (see section 4.1.2.1), this initial state ($\mathcal{P}$) is included in the operator $\mathcal{A}$ which has to be used k times. This state of affairs is not difficult to grasp. A quantum register of n qubits has the capacity of storing $N = 2^n$ states, this corresponding to the exponential storing capability of a quantum register. However, the complexity of a generic state (and therefore of the algorithm responsible for its loading) typically scales with the Hilbert space volume itself, that is to say, exponentially. This is usually referred to as being "inefficient", reserving the attribute "efficient" for cases where the loading complexity is at maximum polynomial in n.

The formulation of the quantum circuit initialization problem in terms of the complexity of the loading algorithm reveals its connections to the theory of complexity and to the theory of approximation. In fact, in order to seek for efficient loading algorithms, we can explore the possibility of approximating the state to be loaded, considering thereby a trade-off between the accuracy and the resources necessary to load the state.

The generic idea of loading an approximated version of the state, aiming to have an efficient process, can take different practical paths. Two classes of strategies are particularly interesting: the "machine learning" approach and the "generation approach". Note that this classification can be handy to clarify the ideas but it is by no means strict, variational circuits –for instance– belong to both classes.

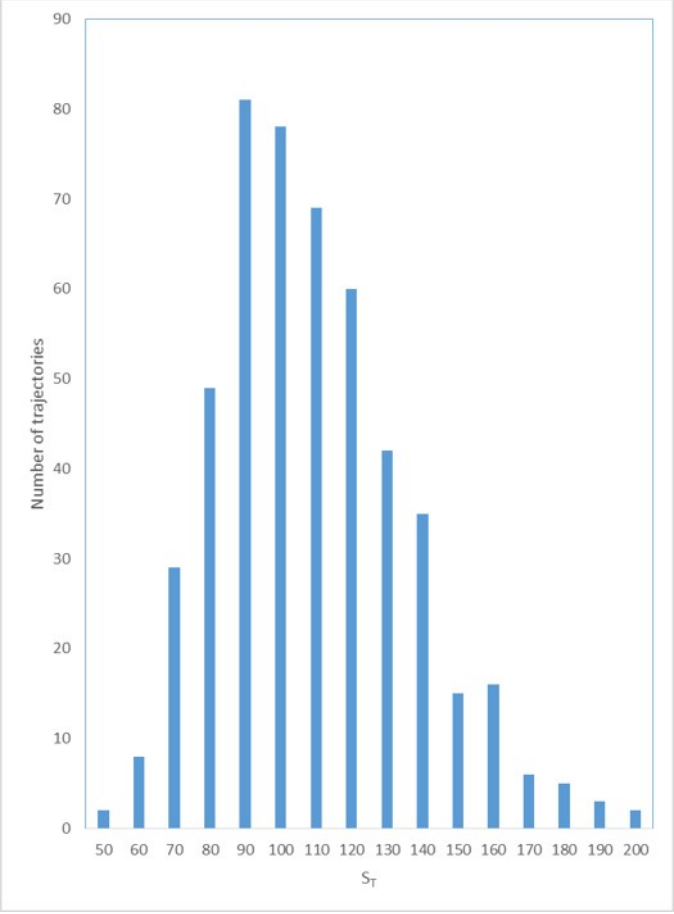


***Figure 6**: Distribution of S_T of 500 trajectories starting at $S_0 = 100$*

In the quantum integration algorithms presented in the previous sections, the first step is to load a probability distribution. Figure 7 shows the distribution of the prices of the HSBC bank in the New York Exchange market from April 2020 to April 2021. This is an example of the distribution to load for VaR calculations,

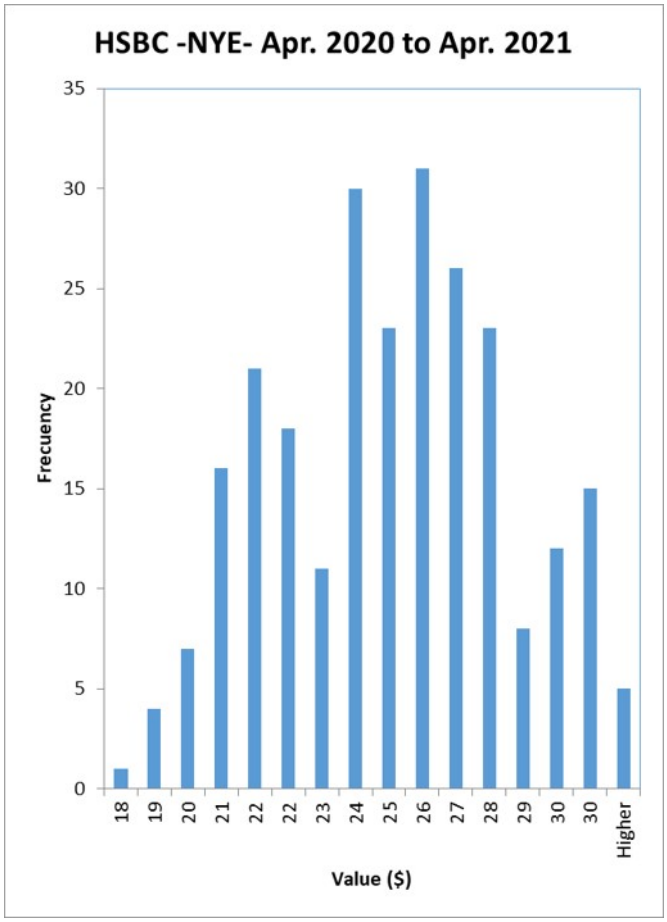


***Figure 7**: Distribution of HSBC share prices from Apr. 2020 to Apr. 2021 in the New York Exchange Market*

which has to achieve a great accuracy. However, for the option pricing, a typical distribution of the final values after simulation of the trajectories looks like Figure 6, which requires less precision for the calculation of the expected values.

In both cases, loading the distribution means that it is necessary to create a circuit for a unitary operation $\mathcal{P}$ such as:

$$|P\rangle = \mathcal{P}|0\rangle \ , \tag{4.106}$$

$$|P\rangle = \sum_{i=0}^{N-1} \sqrt{P_i}|i\rangle \ , \tag{4.107}$$

$$\sum_{i=0}^{N-1} P_i = 1 \ . \tag{4.108}$$

Take into account that $\mathcal{P}$ is diagonal, which can simplify some techniques. In fact, Montanaro (García-Ripoll, 2021; Montanaro, 2015) shows that in QCMC the time of calculation is dominated by the asymptotic time to load the probability distribution as

$$T_{QMC} = O\left(\frac{T_{\mathcal{P}}}{\epsilon}\right) \ , \tag{4.109}$$

being T_{QMC} the time cost of the ideal amplitude estimation algorithm, $T_{\mathcal{P}}$ the time for implementing P and ϵ the desired sampling precision.

There are different approaches to load this distribution:

1. Use a general method to convert unitary operators to circuits (Goubault de Brugière, 2020).

2. Use specific methods to initilize the amplitudes to a normalized vector (Kaye & Mosca, 2004; Shende et al., 2006; Soklakov & Schack, 2006).
3. Use the properties of the probability distribution to create an efficient circuit (L. Grover & Rudolph, 2002).
4. Create an ad-hoc circuit using Parameterized Quantum Circuit (PQC) which approximates the amplitudes (Nakaji et al., 2021).
5. Using Tensor Networks techniques (García-Ripoll, 2021; Holmes & Matsuura, 2020; Ran, 2020) (see Section 4.3.1.3 for further comments).

The first two methods present the problem highlighted at the beginning of this section: the poor scalability. The minimum number of gates to load the distribution has a complexity of $O(N)$, which is too large for the current NISQ computers when the number n of qubits increases. So, the other methods seem to be the right approach to load an approximate probability distribution to the states. We review them in the next subsections.

However, as (Bouland et al., 2020) pointed out, these distributions are only initial prospects of the real pricing computational problem. They are generated using complex Monte Carlo models, that are really the bottleneck in the classical arena. For example, to generate the distribution of Figure 6, hundreds of simulated paths of the value of the asset have been simulated using a simple model, as shown in Figure 8. In order to have a real gain from the usage of Quantum Computers, a method to make such Monte Carlo method with them that keeps the superposition of the underlying evolution model and the payoff is needed, because $\mathcal{P}$ and $\mathcal{R}$ have to be applied several times. If such $\mathcal{P}$ exists, and is fast, it will substitute the computational expensive classical Monte Carlo.

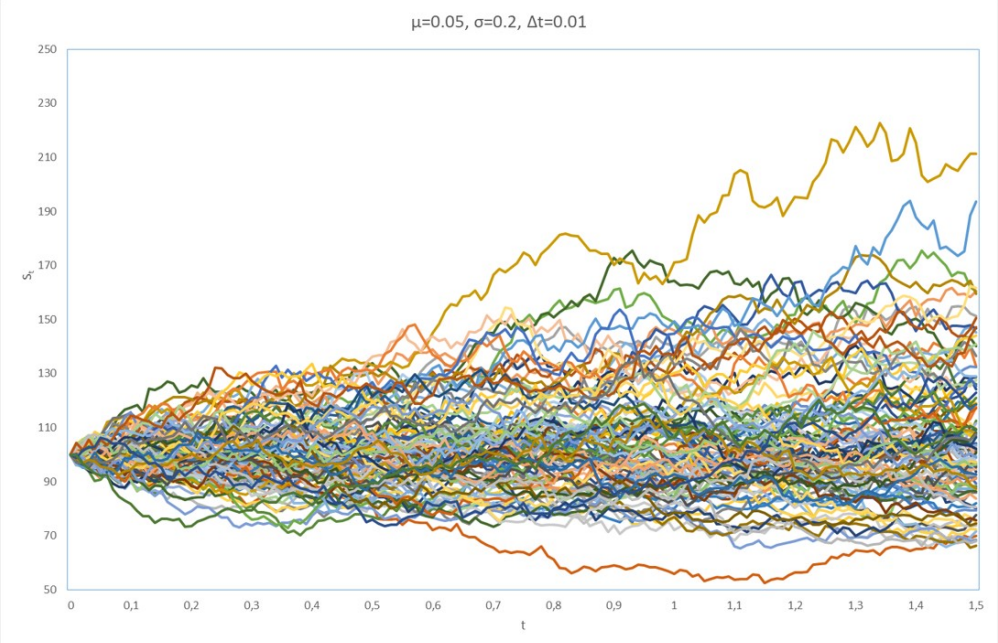


***Figure 8**: Simulated trajectories of one asset using the BS model starting at value 100*

4.3.1.1. Specific methods to load probability distributions

(L. Grover & Rudolph, 2002) proposed a general methodology to load integrable probability distributions into the states efficiently. The basic idea is to discretize it in 2^n regions iteratively, splitting in each step one region in two. In the first step, the initial state is prepared to:

$$|0\rangle \otimes |0\rangle_{n-1} \rightarrow (\sqrt{p_l}|0\rangle + \sqrt{p_r}|1\rangle) \otimes |0\rangle_{n-1} \ , \tag{4.110}$$

where p_r and p_l are the probabilities that x lies on the right or on the left of the middle point, respectively. At each iteration step, one additional qubit is added. This doubles the state space and it is necessary to store the extra information coming from dividing each region into two equally spaced subregions. If x_R^i and x_L^i are the right and left boundaries of a region i, it is necessary to apply a θ_i rotation controlled by the state $|i\rangle$ on the new qubit, where θ_i is given by

$$\theta_i = \arccos\left(\sqrt{f(i)}\right), \quad \text{with} \quad f(i) = \frac{\int_{x_L^i}^{\frac{x_R^i + x_L^i}{2}} p(x)dx}{\int_{x_L^i}^{x_R^i} p(x)dx}, \tag{4.111}$$

where $f(i)$ represents the probability of being to the left of the middle point of the region i conditioned by being in that region.

Summarizing, one full iteration consists in the following passages:

$$\sum_{i=0}^{2^m-1} \sqrt{p_i^{(m)}}|i\rangle_m|0\rangle_{(n-m)}|0\rangle_q \tag{4.112}$$

$$\longrightarrow \sum_{i=0}^{2^m-1} \sqrt{p_i^{(m)}}|i\rangle_m|0\rangle_{(n-m)}|\theta_i\rangle \tag{4.113}$$

$$\longrightarrow \sum_{i=0}^{2^m-1} \sqrt{p_i^{(m)}}|i\rangle_m(\cos(\theta_i)|0\rangle + \sin(\theta_i)|1\rangle)|0\rangle_{(n-m-1)}|\theta_i\rangle \tag{4.114}$$

$$\longrightarrow \sum_{i=0}^{2^m-1} \sqrt{p_i^{(m)}}|i\rangle_m(\cos(\theta_i)|0\rangle + \sin(\theta_i)|1\rangle)|0\rangle_{(n-m-1)}|0\rangle_q \tag{4.115}$$

$$\longrightarrow \sum_{j=0}^{2^{m+1}-1} \sqrt{p_j^{(m+1)}}|j\rangle_{m+1}|0\rangle_{(n-m-1)}|0\rangle_q\,. \tag{4.116}$$

In the first passage, a unitary transformation U_i loads θ_i into the auxiliary register, $U_i|0\rangle_q = |\theta_i\rangle$. The auxiliary register is composed by q qubits and we indicate this explicitly when the register is in its state 0; q corresponds to the precision with which we encode θ_i. Then, a rotation controlled by $|\theta_i\rangle$ encodes the left/right conditioned probabilities for region i into the qubit $m+1$. Finally, the initial U_i operation is uncomputed, thus resulting in a state with the probabilities for each 2^{m+1} regions mapped into the amplitudes of the states. Thus, by iterating the passages from (4.112) to (4.116), the probabilities of the 2^n regions are eventually mapped into the corresponding amplitudes as desired.

The method could consume less resources in the case in which some parallelization strategy is encountered. However, only an approximation to U_i is known that does not scale for $m < 7$ (Kaneko et al., 2020a) and, in fact, some recent works dispute the claim about the speed-up when used for Monte Carlo (Chakrabarti et al., 2020; Herbert, 2021). Similar algorithms which use the conditioned probabilities are presented by (Kaye & Mosca, 2004).

It is important to stress that these methods rely on the knowledge of an analytic expression for the PDF (in general with some further assumptions on it such as log-concavity). When dealing with empirical data, such those ones presented in Fig.6 and Fig.7, the calculation of the conditioned probabilities, (4.111), is straightforward and it can be implemented by means of a simple circuit with controlled R_y gates using binary trees (He et al., 2021; Kerenidis & Prakash, 2017). In both cases, the actual possibility of an overall quantum advantage is still debated in the literature.

4.3.1.2. Parameterized Quantum Circuits

With PQC we loosely refer to all methods which rely on a parametric ansatz whose parameters are optimized by means of a machine learning process (Benedetti et al., 2020). The optimization can be either classical, giving rise to a hybrid algorithm, or quantum and directly embedded into the quantum circuit. At any rate, we are here in the domain of approximation theory tackled with optimization algorithms. Within this wide family we stress two (overlapping) groups of approaches:

1. Variational circuits. In this case, a specific circuit is constructed to approximate the unitary operation U of (4.106). An initial unitary $V(\Theta)$ is created, usually based on basic one and two qubits operations which depend on a set of k parameters $\Theta = \{\theta_1, \ldots, \theta_k\}$. The final values of these parameters are selected by optimization, trying to minimise a cost function as the fidelity or the distance between the initial vector and the final amplitudes. For example, (Nakaji et al., 2021) proposed the algorithm Approximate Amplitude Encoding (AAE) which can load the components of a real-valuated vector into the amplitudes of a quantum state, including their sign. In this case, the complexity of the final quantum algorithm could be of $O(poly(n))$ as desired.

2. Quantum Generative Adversarial Network (QGAN) has been proposed by (Romero & Aspuru-Guzik, 2021) and (Zoufal et al., 2019). It is based on the classical concept proposed by (Goodfellow et al., 2014), where two deep learning models are trained simultaneously using the input data. One model G, called generator, try to generate data that are checked by a second model D (the discriminator). This model try to distinguish if the data come from a real distribution or not. For the quantum case, G is also a PQC which is trained to fake the discriminator, that can be a classical deep learning model. Once the generator is trained, it can be used alone to load the desired distribution in the states. the discriminator could be a quantum circuit or a classical deep learning model. In fact, G is a parametric circuit $G(\Theta)$ that transforms an initial PDF (coined as latent space) to the final and desired one. The initial distribution is not predefined, and can be an uniform, normal, random or whatever. So, the final model is:

$$|P\rangle = G(\Theta)U|0\rangle \tag{4.117}$$

   where U loads the initial distribution and $G(\Theta)$ transforms it to the final one. In the case that the initial distribution could be the uniform one, this can be implemented using a Walsh-Hadamard operation which applies a Hadamard gate to all the qubits. In the case of a random distribution, it can be implemented easily applying random rotations to them. However, if the needed distribution is the standarized normal PDF, the problem is not resolved. For example, Zoufal et al. (Zoufal et al., 2019) used this technique to make an experiment to solve the option pricing problem. In the experiment, the normal PDF had the best Kolmogorov-Smirnov distance. In the case of Romero et al, the circuit has a different mechanism to take into account the latent space. In this case, the initial random numbers (z) are drawn from a classical PDF and are encoded into the circuit using a small encoding part of one-qubit gates. In this case, the final model is:

$$|P\rangle = G(\Theta)U(f(z))|0\rangle \tag{4.118}$$

   However, training a circuit to reproduce a general PDF is not an easy task, even in the classical paradigm for 1-dimension, as shown by (Zaheer et al., 2017).

Using both methods is not a clear path for getting the needed full speed-up, due to the uncertainty created by the training of the models (Chakrabarti et al., 2020). For each data distribution, a training is needed which needs time and consumes resources and it is a hard work. However, one can assume that after having an initial model for one asset, it should not change a lot from one day to another, and only an automatic small retraining is needed daily.

4.3.1.3. Tensor networks

Tensor networks have been developed in condensed matter physics in order to define convenient ansatzes for the ground state of highly-entangled systems. The ansatz is generically expressed in terms of a product of matrices (or tensors) which have different kinds of indexes: *physical* indexes spanning the Hilbert space and *virtual* indexes which are associated with an auxiliary space. The dimensionality and characteristics of the auxiliary space can be adapted to different situations. Intuitively, the auxiliary space helps in disentangling the quantum state providing a more explicit representation which is easy to handle and interpret, though in a bigger space. Through suitable contraction operations on the virtual indexes, the tensor network reduces to the physical quantum state, with only physical indexes.

Tensor networks can be useful to address the problem of initial loading for quantum circuits if one is able to efficiently encode the desired quantum state into a suitable tensor network ansatz. Such encoding would in general entail multi-qubit operations, which can be traded-off with a deeper circuit composed only by 1- and 2-qubit states, see (Ran, 2020).

Matrix product states (MPS) constitute a widely used class of tensor network ansatzes. These have been long studied in the context of quantum computation (Schön et al., 2005) and have been recently claimed to allow for an efficient and scalable encoding of explicit amplitude functions with a high fidelity, relying just on linear-depth quantum circuits (Holmes & Matsuura, 2020).

Tensor network techniques have been studied also in the quantum-inspired realm of classical algorithms. Two relevant example applications are the efficient classical simulation of Shor's factorization algorithm (Dang et al., 2019) and generic multi-variate analysis (García-Ripoll, 2021) (*e.g.* expected value, Fourier transform, interpolation and solving PDE). In fact, in (García-Ripoll, 2021) the author presents an algorithm to efficiently use MPS techniques to encode smooth, differentiable multivariate functions in quantum registers, as PDF are. Although being a promising method, the extension to a general empirical data-based probability as presented in Figure 7 is not, up to our knowledge, known yet and need further research. He extends the case for multi-dimensional encoding.

4.3.1.4. Generative approach, n-nomial trees and quantum walks

With "generative approach" we refer to the actual simulation (or approximated simulation) of the stochastic processes at discrete times which generate the desired distribution. The discrete stochastic approach is particularly important when the underlying asset SDE does not admit an analytical formulation, or when the option is path dependent (thus requiring the knowledge of the distribution of the underlying at intermediate times before maturity). As a simple example, we can consider the Black and Scholes distribution associated to the multiplicative Brownian motion problem (3.5) and generate the distribution from the actual stochastic process by means of an n-nomial tree approximation, see Section 3.1.3.5.

In the class of generative models we find the variational quantum simulation (Endo et al., 2020), implemented –for example– by means of trinomial trees (Kubo et al., 2020).[16] Note that we can see the n-nomial approach as a way to address the quantum numerical solution of a partial differential equation (Fontanela et al., 2021; García-Ripoll, 2021; Gonzalez-Conde et al., 2021) (see Section 4.2.1). In (Kubo et al., 2020) the up/down transition probabilities (3.38) for the trinomial tree approximation are implemented by means of ladder operators. These are built from the cyclic permutation operators, suitably combined with the identity. Thus, the strategy proposed by (Kubo et al., 2020) relies on the linear combination of unitaries (Childs & Wiebe, 2012).

It is relevant to underline that the algorithm in (Kubo et al., 2020) works with the so-called *direct embedding*, where quantum amplitudes encode probabilities instead of encoding their square roots. This is the same embedding considered in the QCoin approach, therefore it is an interesting future perspective to consider the trinomial approach of (Kubo et al., 2020) as a loading circuit for a QCoin integration.

In (Carrera Vazquez & Woerner, 2021) they extend the idea of using the transition probabilities and propose an algorithm to keep not only the final distribution but also the probabilities of each possible path. Further considerations about this approach are described in (Chakrabarti et al., 2020).

Another interesting member of the class is given by quantum walks (Ambainis et al., 2005; An et al., 2020; Magniez et al., 2011; Montanaro, 2015), especially as quantum alternatives to classical Monte Carlo techniques based on multiple-stage Markov chains. For similar ideas aimed at embedding risk models into the quantum circuit, we refer to (Braun et al., 2021).

One interesting (though somewhat speculative) purpose of quantum walks is to provide a model for financial dynamics which is aware of (or just inspired to) the cognitive dynamics which contributes to drive it (as a process of decision making modeled through superpositions and collapses). In this regard, the superposition and interference among quantum states are aimed to model what can be intuitively described as inhibitory and excitatory interactions among possible parallel paths (Orrell, 2020). The co-operative/competitive interactions among paths are possibly a crucial device for a richer modelization of the financial dynamics which is (usually) not taken into account in classical stochastic modelling. In fact, it requires expensive classical parallelizations, these are however natural in a quantum setup.

To clarify the ideas it is useful to focus on the technical differences among classical random walks and quantum walks. The former are stochastic processes where each discretized step is taken according to some random procedure (*e.g.* the toss of a coin). On the contrary, an ideal quantum walk represents a class

[16] (Hao et al., 2019) contains a study on the quantum implementation of the binomial tree approach.

of *deterministic* evolutions for a wave function. Here the stochasticity is given just by the stochastic nature of the eventual quantum measurement, at least in the cases where decoherence and dissipation phenomena are absent. Actually, suitable consideration of decoherence allows to interpolate between classical random walks and quantum walks (Orrell, 2020). This can be understood as follows. The quantum state of a quantum walk has an auxiliary qubit which controls the decision about the step, similarly to a classical coin which determines if the step is taken upward or downward in a binomial classical random walk. The auxiliary qubit is sometimes called a "quantum coin"[17] and it evolves, for instance, with a Hadamard operation at each discrete time step. The Hadamard evolution for the auxiliary qubit represents the quantum version of a fair coin where an up or down state is evenly mapped to up and down at the next step. Yet, we stress once again, the ideal evolution of the quantum state is here deterministic. Unless we consider deviations from ideality and introduce some decoherence and, for instance, go to the extreme non-ideal case in which decoherence is pushed to its maximum, where the "quantum coin" is measured at each time step. In this extreme situation the Hadamard quantum coin reduces to a classical fair coin.

4.3.1.5. Loading the payoff function

The task of computing the expectation value of a payoff function with respect to a probability distribution, see Eq.(4.6), requires that both be loaded into the quantum circuit, accordingly to the pipeline depicted in Fig.2. As already commented below Eq.(4.6), the probability distribution and the payoff function are in general loaded separately. Nevertheless, they can be loaded simultaneously (Carrera Vazquez & Woerner, 2021).

The payoff functions for European vanilla options, see Eq.(3.1), are relatively easy to implement by means of a comparator circuit, as described explicitly in section 4.1.4. As long as the payoff function is piecewise linear, one can generalize the approach implemented for the European vanilla contracts. Nonetheless, this would already entail an increased level of complexity in the quantum circuit; for instance, any separation point between two linear regions would require a comparator circuit.[18]

In a completely general case, *i.e.* for arbitrary payoff, its loading problem can be even more complicated than that of the probability distribution function. In fact, one needs to load the payoff on a quantum state which already encodes the probability distribution. On the contrary, one in general loads the probability distribution starting from a standard reference quantum state (*e.g.* $|0\rangle_n$).

For small size examples, the payoff can be loaded point-wise. However this is clearly an approach which scales inefficiently and cannot be planned for real applications. However in practice is the main approach used.

4.3.2. Scaling to real-world problems

The loading problem commented in Subsection 4.3.1 provides a concrete example about possible issues encountered in designing *full* quantum algorithms able to reach a quantum advantage. An exponential speedup concentrated in a subroutine of an overarching inefficient algorithm, however interesting, is not sufficient to reach quantum advantage.

We are here implicitly referring (as it often happens) to quantum advantage in terms of scaling of the execution time. This is only a part of a bigger picture which needs to involve other variables such as the energy consumption and cost. Strangely enough, this wider picture is usually not analyzed in the quantum finance literature.

Many ideal algorithm studied in the literature are not viable on current or near-future quantum technology. They usually require either an exceedingly large number of qubits or involve too deep a circuit with respect to the realistic coherence time, or both. The theoretical analysis of algorithms should be always accompanied by a critically explored awareness of current and future technological limitations. In this perspective, an important (negative) claim has been described in (Babbush et al., 2021), where it is argued that a quadratic speedup is not sufficient to obtain a quantum advantage, mainly due to the -constant but large- resource overheads (mainly needed for error correction). An important overarching suggestion emerging

[17] Not to be confused with the quantum coin algorithm we have introduced for integration.
[18] See (Cuccaro et al., 2004) for the construction of a quantum comparator circuit as a modification of a ripple adder.

from (Babbush et al., 2021) is that the complexity scaling is in general not enough to properly define an actual threshold for quantum advantage.

In (Chakrabarti et al., 2020) the authors analyze the resources to attain a practically valuable quantum advantage in derivative pricing. They refer to benchmark, path-dependent cases, specifically to autocallable options and target accrual redemption forward contracts. They argue that the complexity of the pricing task implied by path dependence is a necessary ingredient to find a regime where quantum technologies can lead to an advantage with respect to their classical counterparts. However, the benchmark cases are showed to need $7.5k$ logical qubits and a depth of 46 million T-gates and a clock-rate of 10MHz (current quantum technologies moves on the order of 10kHz). These are recognized as markedly prohibitive for the moment, yet they set an order-of-magnitude scenario, useful to frame further research and strategy. An important technical aspect of the paper consists in basing the computation on returns instead of levels of the underlying asset value. This is sometimes necessary, *e.g.* when performances of the underlying assets are defined in terms of returns.

The discussions about realistic implementations of quantum algorithms for finance cannot, at least so far, be addressed in a hardware-independent fashion. Connectivity of the actual computing architecture or even the kind of technology they are based upon are significant factors in discussing the concrete viability of a quantum algorithm.

5. Conclusions

In recent years we have seen significant advances in quantum algorithms with application to financial mathematical problems. While this progress is very encouraging, further work will be required to prove that Quantum Computing can deliver real-world advantage to the areas of derivative pricing and financial risk management. Especially if this advantage ought to be delivered on Noisy Intermediate-Scale Quantum technology with limitations to both the number of logical qubits and the width of quantum circuits.

Recent achievements in Quantum Amplitude Estimation, which can be used to replace classical Monte Carlo approaches, show that the theoretical quadratic speed up can be delivered while avoiding resource-demanding Quantum Fourier Transforms, providing grounds for optimism. A novel candidate algorithm in this regard has been presented here which relies on the QCoin algorithm 4.1.3.

Further theoretical work is needed in order to find efficient (ideally optimal) ways to load probability distributions to quantum registers, as well as efficient mathematical representations of pay-off functions using unitary transforms that can be easily implemented in a quantum circuit.

An area that is also showing some interesting results is the solution of PDEs using quantum computers with applications to derivative pricing and risk management. While exciting, these approaches have not yet been able to prove whether quantum algorithms can provide an advantage over their classical counterparts. Lastly, relatively recent advances in both classical and Quantum Machine Learning algorithms for solving PDEs are also exciting, especially because it has been shown that QML can be robust when implemented in noisy hardware. Furthermore, QML algorithms can present some interesting theoretical advantages over their classical counterparts, see for example (Huang et al., 2021), where the authors introduce the concept of ”projected quantum kernel” to show numerical results that promise a quantum advantage in learning algorithms. These kernels work by projecting the quantum states to an approximate classical representation, helping reduce the dimensionality of the problem, however for real world examples the dimension would be still too large to be handled efficiently using a classical computer. Quantifying quantum advantage for ML algorithms is however not straightforward and should be approached carefully.

In summary, research into financial applications of quantum computing is accelerating with new ideas emerging at rapid pace and while important breakthroughs across the technology stack will be needed to make the approach viable, the recent accelerated publication of important results is encouraging.

List of Acronyms

Term	Definition
ANN	Artificial Neural Network
AAE	Approximate Amplitude Encoding
BSDE	Backward Stochastic Differential Equations
CDF	Cumulative Distribution Function
ChF	Characteristic function
CVaR	Conditional Value-at-Risk
HHL	Harrow, Hassidim and Lloyd algorithm
i.i.d.	independent and identically distributed
MC	Monte Carlo
NEASQC	NExt ApplicationS of Quantum Computing
NISQ	Near Intermediate Scale Quantum
PCA	Principal Component Analysis
PDE	Partial Differential Equation
PDF	Probability Density Function
PQC	Parameterized Quantum Circuit
QFT	Quantum Fourier Transform
QGAN	Quantum Generative Adversarial Network
QCMC	Quantum Computing Monte Carlo
QML	Quantum Machine Learning
QNN	Quantum Neural Network
QPCA	Quantum Principal Component Analysis
QPE	Quantum Phase Estimation
QRAM	Quantum Random Access Memory
RBM	Restricted Boltzmann Machine
RNG	Random number generators
SDE	Stochastic Differential Equation
UC5	Quantum Financial Applications Use Case
VaR	Value-at-Risk

***Table 1**: Acronyms and Abbreviations*

List of Figures

List of Tables

Bibliography

Aaronson, S., & Rall, P. (2020). Quantum approximate counting, simplified. *Symposium on simplicity in algorithms* (pp. 24–32). SIAM.

Abhijith, J., Adedoyin, A., Ambrosiano, J., Anisimov, P., Bärtschi, A., Casper, W., Chennupati, G., Coffrin, C., Djidjev, H., Gunter, D., Karra, S., Lemons, N., Lin, S., Malyzhenkov, A., Mascarenas, D., Mniszewski, S., Nadiga, B., O'Malley, D., Oyen, D., . . . Lokhov, A. Y. (2020). Quantum algorithm implementations for beginners [arXiv:1804.03719].

Abrams, D. S., & Williams, C. P. (2004). Fast quantum algorithms for numerical integrals and stochastic processes [arXiv:quant-ph/9908083].

Adachi, S. H., & Henderson, M. P. (2015). Application of quantum annealing to training of deep neural networks [arXiv:1510.06356].

Alcazar, J., Leyton-Ortega, V., & Perdomo-Ortiz, A. (2020). Classical versus quantum models in machine learning: Insights from a finance application. *Machine Learning: Science and Technology*, *1*(3), 035003.

Ambainis, A., Kempe, J., & Rivosh, A. (2005). Coins make quantum walks faster. *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, 1099–1108.

An, D., Linden, N., Liu, J.-P., Montanaro, A., Shao, C., & Wang, J. (2020). Quantum-accelerated multilevel Monte Carlo methods for stochastic differential equations in mathematical finance [arXiv:2012.06283].

Babbush, R., McClean, J. R., Newman, M., Gidney, C., Boixo, S., & Neven, H. (2021). Focus beyond quadratic speedups for error-corrected quantum advantage. *PRX Quantum*, *2*, 010103.

Beck, C., Hutzenthaler, M., Jentzen, A., & Kuckuck, B. (2021). An overview on deep learning-based approximation methods for partial differential equations [arXiv:2012.12348].

Beer, K., Bondarenko, D., Farrelly, T., Osborne, T. J., Salzmann, R., Scheiermann, D., & Wolf, R. (2020). Training deep quantum neural networks. *Nature Communications*, *11*(808).

Benedetti, M., Lloyd, E., Sack, S., & Fiorentini, M. (2020). Parameterized quantum circuits as machine learning models. *Quantum Science and Technology*, *5*(1), 019601.

Benioff, P. (1980). The computer as a physical system: A microscopic quantum mechanical Hamiltonian model of computers as represented by turning machines. *Journal of Statistical Physics*, *22*(5), 563–591.

Biamonte, J., Wittek, P., Pancotti, N., Rebentrost, P., Wiebe, N., & Lloyd, S. (2017). Quantum machine learning. *Nature*, *549*, 195–202.

Black, F., & Scholes, M. (1973). The pricing of options and corporate liabilities. *Journal of Political Economy*, *81*(3), 637–654.

Black, F. (1976). The pricing of commodity contracts. *Journal of Financial Economics*, *3*(1), 167–179.

Bouland, A., van Dam, W., Joorati, H., Kerenidis, I., & Prakash, A. (2020). Prospects and challenges of quantum finance [arXiv:2011.06492].

Brassard, G., Hoyer, P., Mosca, M., & Tapp, A. (2000). Quantum amplitude amplification and estimation. *AMS Contemporary Mathematics Series*, *305*.

Braun, M. C., Decker, T., Hegemann, N., Kerstan, S. F., & Schäfer, C. (2021). A quantum algorithm for the sensitivity analysis of business risks [arXiv:2103.05475].

Carr, P., & Madan, D. B. (1999). Option valuation using the fast Fourier transform. *Journal of Computational Finance*, *2*, 61–73.

Carrera Vazquez, A., Hiptmair, R., & Woerner, S. (2020). Enhancing the quantum linear systems algorithm using Richardson extrapolation [arXiv:2009.04484].

Carrera Vazquez, A., & Woerner, S. (2021). Efficient state preparation for quantum amplitude estimation. *Physical Review Applied*, *15*, 034027.

Chakrabarti, S., Krishnakumar, R., Mazzola, G., Stamatopoulos, N., Woerner, S., & Zeng, W. J. (2020). A threshold for quantum advantage in derivative pricing [arXiv:2012.03819].

Childs, A. M., & Wiebe, N. (2012). Hamiltonian simulation using linear combinations of unitary operations. *Quantum Information & Computation*, *12*(11–12), 901–924.

Clopper, C. J., & Pearson, E. S. (1934). The use of confidence or fiducial limits illustrated in the case of the binomial. *Biometrika*, *26*(4), 404–413.

Cover, T. M. (2005). *Elements of information theory* (2nd ed.). Wiley.

Coyle, B., Henderson, M., Le, J. C. J., Kumar, N., Paini, M., & Kashefi, E. (2021). Quantum versus classical generative modelling in finance. *Quantum Science and Technology*, *6*(2), 024013.

Cuccaro, S. A., Draper, T. G., Kutin, S. A., & Petrie Moulton, D. (2004). A new quantum ripple-carry addition circuit [arXiv:quant-ph/0410184].
Dang, A., Hill, C. D., & Hollenberg, L. C. L. (2019). Optimising matrix product state simulations of Shor's algorithm. *Quantum*, *3*, 116.
Egger, D. J., Gambella, C., Marecek, J., McFaddin, S., Mevissen, M., Raymond, R., Simonetto, A., Woerner, S., & Yndurain, E. (2020). Quantum computing for finance: State-of-the-art and future prospects. *IEEE Transactions on Quantum Engineering*, *1*, 1–24.
Egger, D. J., Gutiérrez, R. G., Mestre, J. C., & Woerner, S. (2020). Credit risk analysis using quantum computers. *IEEE Transactions on Computers*.
Egger, D. J., & Woerner, S. (2019). Quantum risk analysis. *Quantum Information*, *5*(1).
Endo, S., Sun, J., Li, Y., Benjamin, S. C., & Yuan, X. (2020). Variational quantum simulation of general processes. *Physical Review Letters*, *125*(1).
Fang, F., & Oosterlee, C. W. (2008). A novel pricing method for European options based on Fourier-cosine series expansions. *SIAM Journal on Scientific Computing*, *31*, 826–848.
Feynman, R. P. (1982). Simulating Physics with Computers. *International Journal of Theoretical Physics*, *21*(6/7).
Fontanela, F., Jacquier, A., & Oumgari, M. (2021). A quantum algorithm for linear PDEs arising in finance [arXiv:1912.02753].
García-Molina, P., Rodríguez-Mediavilla, J., & García-Ripoll, J. J. (2021). Solving partial differential equations in quantum computers.
García-Ripoll, J. J. (2021). Quantum-inspired algorithms for multivariate analysis: From interpolation to partial differential equations. *Quantum*, *5*, 431.
Giles, M. B. (2015). Multilevel Monte Carlo methods. *Acta Numerica*, *24*, 259–328.
Giurgica-Tiron, T., Kerenidis, I., Labib, F., Prakash, A., & Zeng, W. (2020). Low depth algorithms for quantum amplitude estimation [arXiv:2012.03348].
Glasserman, P. (2004). *Monte Carlo methods in financial engineering*. Springer.
Gonzalez-Conde, J., Rodríguez-Rozas, Á., Solano, E., & Sanz, M. (2021). Pricing financial derivatives with exponential quantum speedup [arXiv:2101.04023].
Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative adversarial networks [arXiv:1406.2661].
Goubault de Brugière, T. (2020). *Methods for optimizing the synthesis of quantum circuits* (Theses). Université Paris-Saclay.
Grinko, D., Gacon, J., Zoufal, C., & Woerner, S. (2020). Iterative quantum amplitude estimation. *Quantum Information*, *7*(52).
Grover, L., & Rudolph, T. (2002). Creating superpositions that correspond to efficiently integrable probability distributions [arXiv:quant-ph/0208112].
Grover, L. K. (1996). A fast quantum mechanical algorithm for database search. *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, 212–219.
Hao, W., Lefèvre, C., Tamturk, M., & Utev, S. (2019). Quantum option pricing and data analysis. *Quantitative Finance and Economics*, *3*(3), 490–507.
Harrow, A. W., Hassidim, A., & Lloyd, S. (2009). Quantum algorithm for linear systems of equations. *Physical Review Letters*, *103*, 150502.
He, C., Li, J., Liu, W., & Wang, Z. J. (2021). A low complexity quantum principal component analysis algorithm [arXiv:2010.00831].
Herbert, S. (2021). The problem with Grover-Rudolph state preparation for quantum Monte Carlo [arXiv:2101.02240].
Heston, S. (1993). A closed-form solution for options with stochastic volatility with applications to bond and currency options. *Review of Financial Studies*, *6*, 327–343.
Hirofumi Nishi, T. K., & Matsushita, Y.-i. (2020). Implementation of quantum imaginary-time evolution method on nisq devices: Nonlocal approximation.
Hoeffding, W. (1963). Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, *58*(301), 13–30.
Holmes, A., & Matsuura, A. Y. (2020). Efficient quantum circuits for accurate state preparation of smooth, differentiable functions [arXiv:2005.04351].
Huang, H.-Y., Broughton, M., Mohseni, M., Babbush, R., Boixo, S., Neven, H., & McClean, J. R. (2021). Power of data in quantum machine learning.
Hull, J. (1997). *Options, futures, and other derivatives* (3. ed., internat. ed). Prentice Hall.

Job, J., & Adachi, S. (2020). Systematic comparison of deep belief network training using quantum annealing vs. classical techniques [arXiv:2009.00134].

Kaneko, K., Miyamoto, K., Takeda, N., & Yoshino, K. (2020a). Quantum pricing with a smile: Implementation of local volatility model on quantum computer [arXiv:2007.01467].

Kaneko, K., Miyamoto, K., Takeda, N., & Yoshino, K. (2020b). Quantum speedup of Monte Carlo integration in the direction of dimension and its application to finance [arXiv:2011.02165].

Kaye, P., & Mosca, M. (2004). Quantum networks for generating arbitrary quantum states [arXiv:quant-ph/0407102].

Kerenidis, I., & Prakash, A. (2017). Quantum recommendation systems. In C. H. Papadimitriou (Ed.), *8th innovations in theoretical computer science conference (itcs 2017)* (49:1–49:21). Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

Kloeden, P. E., & Platen, E. (2013). *Numerical solution of stochastic differential equations*. Springer Science & Business Media.

Kubo, K., Nakagawa, Y. O., Endo, S., & Nagayama, S. (2020). Variational quantum simulations of stochastic differential equations [arXiv:2012.04429].

Kyriienko, O., Paine, A. E., & Elfving, V. E. (2021). Solving nonlinear differential equations with differentiable quantum circuits. *Physical Review A*, *103*(5).

Lin, J., Bao, W.-S., Zhang, S., Li, T., & Wang, X. (2019). An improved quantum principal component analysis algorithm based on the quantum singular threshold method. *Physics Letters A*, *383*(24), 2862–2868.

Lloyd, S., Mohseni, M., & Rebentrost, P. (2014). Quantum principal component analysis. *Nature Physics*, *10*, 631–633.

Longstaff, F. A., & Schwartz, E. S. (2001). Valuing American options by simulation. A simple least-squares approach. *Review of Financial Studies*, *14*, 113–147.

Magniez, F., Nayak, A., Roland, J., & Santha, M. (2011). Search via quantum walk. *SIAM Journal on Computing*, *40*(1), 142–164.

Manin, Y. (1980). Computable and Uncomputable (in Russian). *Sovetskoye Radio*.

McArdle, S., Jones, T., Endo, S., Li, Y., Benjamin, S. C., & Yuan, X. (2019). Variational ansatz-based quantum simulation of imaginary time evolution. *Quantum Information*, *5*(75).

McNeil, A. J., Frey, R., & Embrechts, P. (2015). *Quantitative risk management: concepts, techniques and tools* (revised). Princeton University Press.

Merton, R. C. (1974). On the pricing of corporate debt: The risk structure of interest rates. *The Journal of Finance*, *29*(2), 449–470.

Mikosch, T. (1998). *Elementary stochastic calculus : With finance in view*. World Scientifc.

Montanaro, A. (2015). Quantum speedup of Monte Carlo methods. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *471*(2181), 20150301.

Nakaji, K., Uno, S., Suzuki, Y., Raymond, R., Onodera, T., Tanaka, T., Tezuka, H., Mitsuda, N., & Yamamoto, N. (2021). Approximate amplitude encoding in shallow parameterized quantum circuits and its application to financial market indicator [arXiv:2103.13211].

Orrell, D. (2020). A quantum walk model of financial options. *Econometric Modeling: Capital Markets - Asset Pricing eJournal*.

Ortiz-Gracia, L., & Oosterlee, C. W. (2016). A highly efficient Shannon wavelet inverse Fourier technique for pricing European options. *SIAM Journal on Scientific Computing*, *38*(1), 118–143.

Orús, R., Mugel, S., & Lizaso, E. (2019). Quantum computing for finance: Overview and prospects. *Reviews in Physics*, *4*, 1–13.

Preskill, J. (2018). Quantum computing in the NISQ era and beyond. *Quantum*, *2*, 79.

Ramos-Calderer, S., Pérez-Salinas, A., García-Martín, D., Bravo-Prieto, C., Cortada, J., Planagumà, J., & Latorre, J. I. (2020). Quantum unary approach to option pricing [arXiv:1912.01618].

Ran, S.-J. (2020). Encoding of matrix product states into quantum circuits of one- and two-qubit gates. *Physical Review A*, *101*, 032310.

Rebentrost, P., Gupt, B., & Bromley, T. R. (2018). Quantum computational finance: Monte Carlo pricing of financial derivatives. *Physical Review A*, *98*(2).

Reddy, P., & Bhattacherjee, A. B. (2021). A hybrid quantum regression model for the prediction of molecular atomization energies. *Machine Learning: Science and Technology*, *2*(2), 025019.

Romero, J., & Aspuru-Guzik, A. (2021). Variational quantum generators: Generative adversarial quantum machine learning for continuous distributions. *Advanced Quantum Technologies*, *4*(1), 2000003.

Scholz, F. (2008). Confidence bounds and intervals for parameters relating to the binomial, negative binomial, poisson and hypergeometric distributions with applications to rare events.

Schön, C., Solano, E., Verstraete, F., Cirac, J. I., & Wolf, M. M. (2005). Sequential generation of entangled multiqubit states. *Physical Review Letters*, *95*(11).
Schuld, M. (2021). Supervised quantum machine learning models are kernel methods.
Schuld, M., Sinayskiy, I., & Petruccione, F. (2016). Prediction by linear regression on a quantum computer. *Physical Review A*, *94*, 022342.
Shao, C. (2020). Data classification by quantum radial-basis-function networks. *Physical Review A*, *102*, 042418.
Shende, V., Bullock, S., & Markov, I. (2006). Synthesis of quantum-logic circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, *25*(6), 1000–1010.
Shimada, N. H., & Hachisuka, T. (2020). Quantum coin method for numerical integration [arXiv:1910.00263].
Shor, P. (1994). Algorithms for quantum computation: Discrete logarithms and factoring. *Proceedings 35th Annual Symposium on Foundations of Computer Science*, 124–134.
Soklakov, A. N., & Schack, R. (2006). Efficient state preparation for a register of quantum bits. *Physical Review A*, *73*, 012307.
Stamatopoulos, N., Egger, D. J., Sun, Y., Zoufal, C., Iten, R., Shen, N., & Woerner, S. (2020). Option pricing using quantum computers. *Quantum*, *4*, 291.
Suzuki, Y., Uno, S., Raymond, R., Tanaka, T., Onodera, T., & Yamamoto, N. (2020). Amplitude estimation without phase estimation. *Quantum Information Processing*, *19*(2).
Tan, K. C. (2020). Fast quantum imaginary time evolution.
Vázquez, C. (2010). An introduction to Black-Scholes modeling and numerical methods in derivatives pricing. *MAT-Serie A, Universidad Austral*, *17*.
Vinci, W., Buffoni, L., Sadeghi, H., Khoshaman, A., Andriyash, E., & Amin, M. H. (2020). A path towards quantum advantage in training deep generative models with quantum annealers. *Machine Learning: Science and Technology*, *1*(4), 045028.
Wang, G. (2017). Quantum algorithm for linear regression. *Physical Review A*, *96*, 012335.
Wiebe, N., Braun, D., & Lloyd, S. (2012). Quantum algorithm for data fitting. *Physical Review Letters*, *109*, 050505.
Wilmott, P. (2007). *Paul Wilmott introduces quantitative finance* (2nd ed.). Wiley-Interscience.
Xin, T., Che, L., Xi, C., Singh, A., Nie, X., Li, J., Dong, Y., & Lu, D. (2021). Experimental quantum principal component analysis via parametrized quantum circuits. *Physical Review Letters*, *126*, 110502.
Zaheer, M., Li, C.-l., Póczos, B., & Salakhutdinov, R. (2017). GAN connoisseur: Can GANs learn simple 1D parametric distributions?
Zhao, Z., Fitzsimons, J. K., & Fitzsimons, J. F. (2019). Quantum-assisted Gaussian process regression. *Physical Review A*, *99*, 052331.
Zoufal, C., Lucchi, A., & Woerner, S. (2019). Quantum generative adversarial networks for learning and loading random distributions. *Quantum Information*, *7*(103).

