Archives of Computational Methods in Engineering (2022) 29:4137–4163
https://doi.org/10.1007/s11831-022-09732-9

REVIEW ARTICLE

A Survey on Quantum Computational Finance for Derivatives Pricing and VaR

Andrés Gómez[1] · Álvaro Leitao[2] · Alberto Manzano[2] · Daniele Musso[1,4] · María R. Nogueiras[3] · Gustavo Ordóñez[3] · Carlos Vázquez[2]

Received: 9 September 2021 / Accepted: 28 February 2022 / Published online: 28 March 2022

Abstract
We review the state of the art and recent advances in quantum computing applied to derivative pricing and the computation of risk estimators like Value at Risk. After a brief description of the financial derivatives, we first review the main models and numerical techniques employed to assess their value and risk on classical computers. We then describe some of the most popular quantum algorithms for pricing and VaR. Finally, we discuss the main remaining challenges for the quantum algorithms to achieve their potential advantages.

1 Introduction

Quantum computing for financial applications is a hot topic, with multiple and varied possibilities [14, 70]. The main reason for exploring this field of application for quantum computing comes from the fact that current algorithms used by financial institutions demand high-performance computing, as most of the models used in mathematical finance do not have analytical solutions. For an increasing number of cases, quantum computing promises faster responses that can provide solutions to up-to-now intractable problems.

In this article, the set of algorithms to be explored corresponds to the pricing of financial derivatives and Value-at-Risk (VaR) computation.

In the first case, the main objective is to provide the value of an option, which is a financial derivative that gives the right to sell or buy an asset in a future date at a prescribed price. Financial derivatives are contracts whose value depends on the value of a particular underlying asset. In the second case, the VaR is a risk measure so that the objective is to forecast the losses of future operations. The present article reviews the main concepts and classical algorithms to address both mathematical problems, as well as the recent advances to solve these problems with quantum computing.

The article is divided into two main parts. The first one, Sect. 2: "Classical methods in pricing and VaR", provides a quick overview of the approaches currently used to price and manage the risk of financial derivatives. 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

Andrés Gómez, Álvaro Leitao, Alberto Manzano, Daniele Musso, María R. Nogueiras, Gustavo Ordóñez and Carlos Vázquez have contributed equally to this work.

✉ Carlos Vázquez
carlos.vazquez.cendon@udc.es

Andrés Gómez
andres.gomez.tato@cesga.es

Álvaro Leitao
alvaro.leitao@udc.gal

Alberto Manzano
alberto.manzano.herrero@udc.es

Daniele Musso
mussodaniele@uniovi.es

María R. Nogueiras
maria.r.nogueiras@hsbc.com

Gustavo Ordóñez
gustavo.ordonez@moodys.com

[1] Galician Supercomputing Center (CESGA), Santiago, Spain

[2] Department of Mathematics and CITIC, Universidade da Coruña, A Coruña, Spain

[3] Global Risk Analytics, HSBC, London, UK

[4] Department of Physics and Instituto de Ciencias y Tecnologías Espaciales de Asturias (ICTEA), Oviedo University, Oviedo, Spain

agreed today. This means that, if at that future date (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.

Section 2 starts with a short review of some basic option contracts and their payoff functions. It continues with some modelling approaches and finally covers two general numerical approaches for derivatives pricing: Monte Carlo and Partial Differential Equations- PDEs (including machine learning approaches). In the second part of Sect. 2, 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 Expected Shortfall). Factor models, a key dimension reduction technique, 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 article, Sect. 3: "Quantum methods in pricing and VaR", contains a review of the currently known approaches to implement derivative pricing and derivative risk management on quantum computers.

Section 3 starts discussing quantum alternatives to classical Monte Carlo. The cornerstone of the quantum alternatives to Monte Carlo rely on Quantum Amplitude Estimation and more modern variations on this technique. 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 dimensionality", that arises when the number of underlying assets or stochastic factors becomes large. Later on, still in Sect. 3, quantum methods for solving derivative pricing models using PDEs are also discussed. In the third block of Sect. 3 we introduce some numerical quantum techniques for pricing and VaR which, at least in spirit, are closer to Machine Learning. Section 3 concludes with a review of the key remaining challenges for quantum alternatives to Monte Carlo to deliver their potential advantage, namely, the efficient loading or generation of probability distributions in a quantum computer and the computation of the payoff functions in pricing problems.

2 Classical Methods for Pricing and VaR

2.1 Pricing of Financial Derivatives

2.1.1 Financial Derivatives and Options

In financial markets, a *derivative* is a financial contract whose value depends on the future performance of one or several *assets* (usually referred to as *underlying asset*, or simply *underlying*). Examples of underlying assets are stocks, interest rates, exchange rates, credit spreads, etc. 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. Let us denote by v_t and S_t the value of the option and the underlying at time t, respectively. At the expiring date of the option contract T, or before under certain circumstances, the option holder receives a payment, referred as *payoff*, that depends on the evolution of the underlying asset. 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 value of the underlying at the expiry date S_T. There also exist options with payoffs depending on previous values of the asset (exotic options). At any time $0 < t < T$, the option value is given by $v_t = V(t, S_t)$, where V is a function depending on time t and the value of the underlying S_t. The option value 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 the underlying asset and option *fair* values, i.e., the premium which makes both the option seller and buyer break even.[1]

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

European Options A European vanilla option on an 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) [50]. When the option confers the right to buy it is referred to as a *call option*, and when it gives the right to sell it is called a *put option*.

The payoffs of the European call and put options are respectively given by

$$c_T = \max\left(S_T - K, 0\right), \quad p_T = \max\left(K - S_T, 0\right), \tag{1}$$

so that $h(T, S_T) = \max\left(S_T - K, 0\right)$ or $h(T, S_T) = \max\left(S_T - K, 0\right)$.

Notice that at maturity date $t = T$ these values are known, since S_T is known and the strike price K is fixed in the contract. The option pricing problem is to calculate the fair value of the contracts at the pricing date $t = 0$, i.e., c_0 and p_0.

For example, in Fig. 1 options on HSBC stock with expiry date April 30th, 2021 are shown.[2]. The value of the HSBC stock at that date was 29.16, The call options marked in blue

[1] Fair value of an option is also defined as a model-derived estimate of the option value in an efficient market.

[2] Data from Yahoo Finance, consulted on Apr. 2nd, 2021.

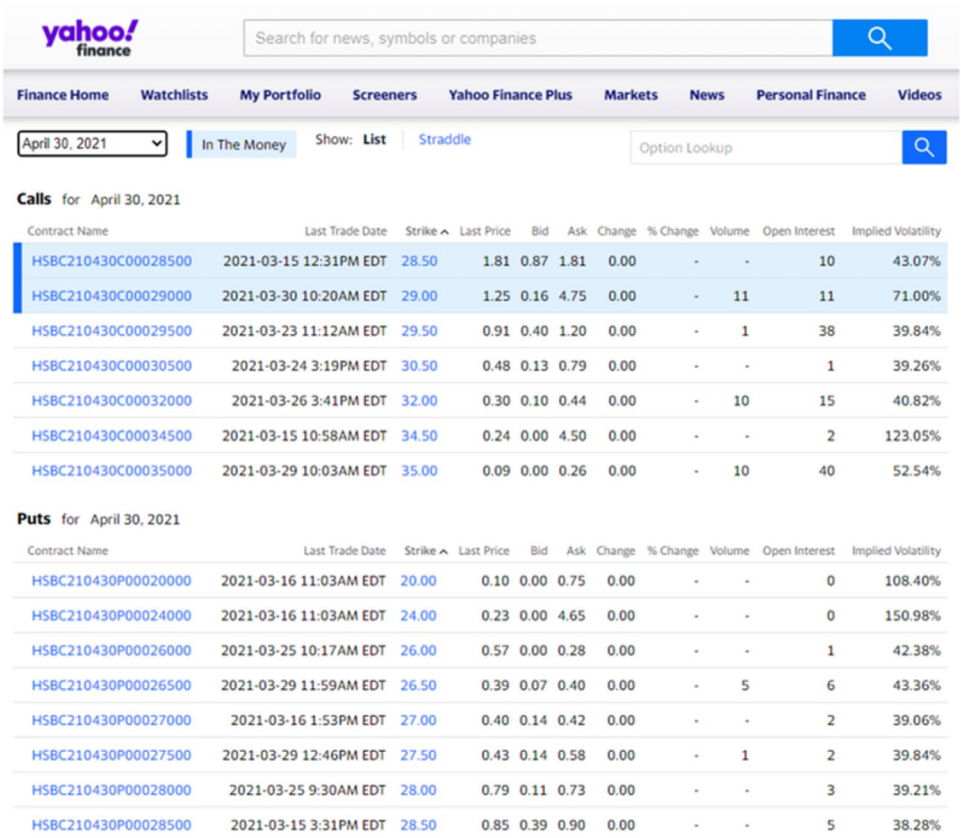

Fig. 1 Options on HSBC stocks (the underlying assets) in Yahoo Financials for expiry date of Apr. 30th, 2021

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 Fig. 1, the *last price* column refers to the most recent price of the option in the market.

American Options Unlike European options, where the option can only be exercised at expiry date T and receive the payoff (1), American options give the holder the right to exercise the call (put) option, *i.e.* the right to buy (or sell) the underlying stock at the strike value at any time $t_e \in [0, T]$ and receive:

$$c_{t_e} = \max\left(S_{t_e} - K, 0\right), \quad p_{t_e} = \max\left(K - S_{t_e}, 0\right), \tag{2}$$

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

$$c_t \geq \max\left(S_t - K, 0\right), \quad p_t \geq \max\left(K - S_t, 0\right), \tag{3}$$

otherwise there would be arbitrage opportunities[3], 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.

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:

$$V_T = f(T, S_T^1, \ldots, S_T^d). \tag{4}$$

Some examples of specific payoffs for basket options are:

[3] Which means opportunities of sure gain without risk, the lack of arbitrage opportunities being one of the main assumptions in option pricing theory

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

where g is a generic function which is applied to the maximum (minimum) of the different assets values in the "Best of" ("Worst of") option, respectively.

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 on vanilla contracts only. For vanilla options, the payoffs only depend on the value of the underlying at expiry date T. More complicated examples are Asian options, where the payoff is determined by certain average of the underlying asset values in a set of discrete times or in a certain period; or barrier options, where the payoff depends on whether the underlying value has or has not crossed some prescribed limits (for further details, see [87], for example). Here we will not consider these non-vanilla options (also called exotic options).

2.1.2 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 and Scholes [13] and Heston dynamics [46] for the underlying asset evolution. This framework can easily be extended to the values of a set of assets when dealing with basket options.

Black–Scholes Model The Black–Scholes model allows us 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 [87]. In the Black–Scholes model one assumes that the stochastic dynamics of the underlying value S_t follows a geometric Brownian motion, so that the stochastic process S_t satisfies the following Stochastic Differential Equation (SDE):

$$dS_t = \mu S_t dt + \sigma S_t dW_t, \tag{5}$$

where μ is the drift parameter, σ is the stock volatility and dW_t represents the increment of a Wiener process, also called a Brownian motion (see [66] for its definition). 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 option value is given by $v_t = V(t, S_t^1, \dots, S_t^d)$ and we can assume the Black-Scholes dynamics for each underlying asset:

$$dS_t^i = \mu^i S_t^i dt + \sigma^i S_t^i dW_t^i, \tag{6}$$

where μ^i, σ^i and W_t^i are the corresponding drift, volatility and driving Brownian motion of asset i, for $i = 1, \dots, d$. 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 coefficients can be time-dependent.

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 Black–Scholes model. Considering two separate processes, one for the value S_t and another one for the instantaneous variance v_t, this provides us with richer dynamics that can fit observed market prices better. One of the most popular stochastic volatility models is the Heston model, that assumes the following dynamics for the underlying asset and its variance:

$$\begin{aligned}
dS_t &= \mu S_t dt + \sqrt{v_t} S_t dW_t^S,\\
dv_t &= \kappa(\theta - v_t) dt + \chi \sqrt{v_t} dW_t^v.
\end{aligned} \tag{7}$$

In the first equation, μ is the drift parameter and $\sqrt{v_t}$ 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^v are considered with correlation coefficient ρ that captures the dependency between asset value and variance, *i.e.* $dW_t^S dW_t^v = \rho\, dt$.

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

2.1.3 Numerical Methods for Options

In order to compute the value of European options with one underlying asset, we can use Girsanov's theorem[4] to work in the risk neutral probability measure Q. Mathematical models for options pricing must take into account the absence of arbitrage hypothesis. For this purpose, the discounted prices of financial products must satisfy the martingale property under certain probability measure, this measure is the so called risk neutral probability measure Q. The dynamics of the underlying asset under this measure is obtained by replacing the drift μ by the risk free rate r in (5). Under the measure Q, the value of the option is a martingale process, that is, the conditional expected value of the discounted

[4] 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 [66].

value of the derivative is constant through time. Next, we use Ito's lemma[5] 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:

$$V(t, S_t) = e^{-r(T-t)}\mathbb{E}_Q[h(T, S_T)|\mathcal{F}_t], \tag{8}$$

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 (given by (1) in European call and put options) 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 [66].

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:

$$v_t = V(t, S_t) = \max_{\tau\in[t,T]} \exp -r(T-\tau)\mathbb{E}_Q[h(\tau, S_\tau)|\mathcal{F}_t], \tag{9}$$

where τ denotes a stopping time.

The pricing of European options admits an equivalent formulation in terms of Partial Differential Equations (PDEs). This formulation can be obtained by using 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's lemma, the building of a risk free portfolio and the consideration of the 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 $\Omega = (0, T) \times \mathbb{R}^+$:

$$\frac{\partial V}{\partial t} + rS\frac{\partial V}{\partial S} + \frac{\sigma^2 S^2}{2}\frac{\partial^2 V}{\partial S^2} - rV = 0, \tag{10}$$

$$V(T, S) = h(T, S). \tag{11}$$

The final condition at time T is given by the function h and depends on the payoff of the specific product. The solution for this PDE is [91]

$$V(t, S) = \frac{e^{-r(T-t)}}{\sigma\sqrt{2\pi(T-t)}}\int_0^\infty \exp\left(-\frac{(\log(S/S') + (r - \frac{\sigma^2}{2})(T-t))^2}{2\sigma^2(T-t)}\right)h(t, S')\frac{dS'}{S'}. \tag{12}$$

Note that for the case of European vanilla call or put options, the solution of (10) has an analytical expression, also known as Black-Scholes formulas:

$$V_c(t, S_t) = S_t\mathcal{N}(0, 1)(d_1) - Ke^{-r(T-t)}\mathcal{N}(0, 1)(d_2), \tag{13}$$

$$V_p(t, S_t) = -S_t\mathcal{N}(0, 1)(-d_1) + Ke^{-r(T-t)}\mathcal{N}(0, 1)(-d_2), \tag{14}$$

with

$$d_1 = \frac{\log(S/K) + (r + \frac{1}{2}\sigma^2)(T-t)}{\sigma\sqrt{T-t}}, \tag{15}$$

$$d_2 = \frac{\log(S/K) + (r - \frac{1}{2}\sigma^2)(T-t)}{\sigma\sqrt{T-t}} = d_1 - \sigma\sqrt{T-t}, \tag{16}$$

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

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

Monte Carlo The expectation in Eq. (8) can be written in integral form as a *risk-neutral valuation formula*,

$$V(t, S) = e^{-r(T-t)}\int_{\mathbb{R}} h(T, y)f(y|\mathcal{F}_t)\,dy\,, \tag{17}$$

with $f(\cdot)$ being the Probability Density Function (PDF) of the asset value process S_t.

Many numerical techniques are based on the solution provided by this risk-neutral valuation formula, in particular by taking advantage of its integral formulation.

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, $X_1, X_2, \ldots, X_M$. We then define a Monte Carlo estimator (see [35], for further details) as

[5] 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.

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

If g is integrable over C then, by the *strong law of the large numbers*, $\bar{I}_M \to I$ as $M \to \infty$, 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 $s_g/\sqrt{M}$. Therefore, the quadratic error of the Monte Carlo estimate tends to zero with order $\mathcal{O}(1/\sqrt{M})$ when M tends to infinity, 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 readily extendable to multi-dimensional problems without increasing the rate of convergence. Moreover, the simplicity of Monte Carlo methods allows for different convergence improvement techniques, such as *variance reduction techniques* [35] or *multi-level Monte Carlo* acceleration [32].

When employing Monte Carlo to approximate an expectation value, an essential part of the method is the 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 the context of the Itô 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 [56] for further details on numerical methods for SDE.

In order to use Eq. (8) the expectation of S_t at time T needs to be estimated. To obtain this, it is possible to simulate different scenarios of the evolution of the values S_t from Eq. (5) in the Black–Scholes dynamics. These simulations of the values S_t until time T can be obtained by discretizing Eq. (5) with an Euler–Maruyama scheme, starting from the given value S_0 at $t = 0$:

$$S_{j+1}^m = S_j^m + rS_j^m \Delta t + \sigma S_j^m dW_j, \tag{18}$$

$\Delta t = T/(J+1)$ is the uniform time step, $j = 0, \ldots, J+1$ is the index for each time step and $m = 1, \ldots, M$ 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 M simulations of the asset values evolution have been obtained, it is possible to approximate the option value at $t = 0$ in expression (8) by:

$$v_0 = V(0, S_0) \approx \exp(-rT)\,\frac{1}{M}\sum_{m=1}^{M} h(t_{J+1}, S_{J+1}^m). \tag{19}$$

It is straightforward to extend Monte Carlo methods to the case of European options on a finite set of correlated assets following the Black–Scholes dynamics (6). This can be achieved by computing the trajectories of all the 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 to 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. A common approach for pricing American options using the expectation given by expression (9) involves a combination of Monte Carlo and regression techniques. Longstaff-Schwartz is a classical example of these methods [61].

PDEs: Finite Differences A common technique to solve PDEs like (10) 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 $(t_j, S_i) = (j\Delta t, i\Delta S)$, $j = 0, \ldots, J+1$; $i = 0, \ldots, I+1$. At the mesh nodes, the derivatives involved in the PDE (10) are approximated as follows:

$$\frac{\partial V}{\partial S}(t_j, S_i) \approx \frac{V(t_j, S_{i+1}) - V(t_j, S_{i-1})}{2\Delta S}, \tag{20}$$

$$\frac{\partial^2 V}{\partial S^2}(t_j, S_i) \approx \frac{V(t_j, S_{i+1}) - 2V(t_j, S_i) + V(t_j, S_{i-1})}{(\Delta S)^2}, \tag{21}$$

$$\frac{\partial V}{\partial t}(t_j, S_i) \approx \frac{V(t_{j+1}, S_i) - V(t_j, S_i)}{\Delta t}. \tag{22}$$

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 Eq. (10) can be written as the following set of I linear equations for $j = J, J-1, \ldots, 0$:

$$\begin{aligned}
&\frac{V_{j+1,i} - V_{j,i}}{\Delta t} = \\
&\frac{1}{2}\left(-rS_i\frac{V_{j,i+1} - V_{j,i-1}}{2\Delta S} - \frac{\sigma^2 S_i^2}{2}\frac{V_{j,i+1} - 2V_{j,i} + V_{j,i-1}}{(\Delta S)^2} + rV_{j,i}\right) \\
&+ \frac{1}{2}\left(-rS_i\frac{V_{j+1,i+1} - V_{j+1,i-1}}{2\Delta S} - \frac{\sigma^2 S_i^2}{2}\frac{V_{j+1,i+1} - 2V_{j+1,i} + V_{j+1,i-1}}{(\Delta S)^2} + rV_{j+1,i}\right), \\
&i = 1, \ldots, I.
\end{aligned} \tag{23}$$

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:

$$V_{j,I+1} = S_\infty - e^{-r(T-t_j)}K, \qquad V_{j,0} = 0. \tag{24}$$

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 straightforward but it carries the cost of increasing the 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 the PDE problem (10) 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 [87] and the references therein):

$$\mathcal{L}(V) \le 0, \quad V \ge h, \quad \mathcal{L}(V)\cdot(V-h) = 0. \tag{25}$$

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:

$$AV_j \le b_j, \quad V_j \ge h_j, \quad (AV_j - b_j)^t \cdot (V_j - h_j) = 0, \tag{26}$$

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 (26) can be expressed as a quadratic optimization with inequality constraints of the form:

$$V_j = \arg\min_{Y \ge h_j}\left(\frac{1}{2}Y^t AY - b_j^t Y\right). \tag{27}$$

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

PDEs: Artificial Neural Network (ANN) and Deep Learning Unsupervised deep learning techniques can be used in derivative pricing, see [9] and the references therein, for applications for solving both linear and nonlinear time-dependent PDEs such as the ones presented above.

A general PDE problem can be written as:

$$\begin{aligned}
&\mathcal{N}_I(v(t,x)) = 0, \quad x \in \widetilde{\Omega},\ t \in [0,T], \\
&\mathcal{N}_B(v(t,x)) = 0, \quad x \in \partial\widetilde{\Omega},\ t \in [0,T], \\
&\mathcal{N}_0(v(t^*,x)) = 0, \quad x \in \widetilde{\Omega} \text{ and } t^* = 0 \text{ or } t^* = T,
\end{aligned} \tag{28}$$

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 $\widetilde{\Omega}$.

PDEs problems for European and American option pricing can be cast in (28) formulation. In this setting, 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.*,

$$\arg\min_{v\in\mathcal{C}^k} L(v) = \hat{v}\,, \tag{29}$$

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

The solution of (28) can be approximated with a deep neural network and the accuracy of this approximation can be related to the value of the cost function used. The deep neural network consists of an input layer with d neurons, several hidden layers and an output 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 is defined as follows:

$$\begin{aligned} L(v) = \lambda \int_{\Omega} |\mathcal{N}_I(v(t,x))|^p dxdt \\ + (1-\lambda)\int_{\partial\Omega} \left(|\mathcal{N}_B(v(t,x))|^p + |\mathcal{N}_0(v(t,x))|^p\right) dxdt, \end{aligned} \tag{30}$$

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

$$\begin{aligned} &\mathcal{N}_I(v(t,x)) \equiv N(v(t,x)) - F(t,x) \quad \text{in } \Omega\,,\\ &\mathcal{N}_B(v(t,x)) \equiv B(v(t,x)) - G(t,x) \quad \text{on } \partial\widetilde{\Omega}\,,\\ &\mathcal{N}_0(v(t^*,x)) \equiv H(x) - v(t^*,x) \quad \text{in } \widetilde{\Omega}\times t^*, \text{ with } t^*=0 \text{ or } t^*=T, \end{aligned} \tag{31}$$

where N, F, B, G and H are generic functions whose expression depends on the problem at hand. Moreover, the parameter $\lambda \in (0,1)$ in the cost function (30) represents the relative importance of the interior and boundary functions in the minimization process. Also the choice of p depends on the assumed regularity of the solution of the PDE (usually $p=2$ is taken).

Trinomial Trees The trinomial tree scheme allows us to simulate the SDE (5) under the risk-neutral measure [57]. In order to do so, we discretize the time as $t_j = j\Delta t$ with $j = 0, 1, \ldots, J$ and the underlying as $s_i = i\Delta s$ with $i = 0, 1, \ldots, I$. Here $t_J = T$ and $s_I = s_\infty$.

Next, we define the transition probabilities as:

$$\begin{aligned} p_u(s,t) &= \Pr[S_{t+\Delta t} = s + \Delta s | S_t = s],\\ p_m(s,t) &= \Pr[S_{t+\Delta t} = s | S_t = s],\\ p_d(s,t) &= \Pr[S_{t+\Delta t} = s - \Delta s | S_t = s]. \end{aligned} \tag{32}$$

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:

$$\begin{aligned} p_u(s_i,t_j) &= \frac{1}{2}\left(\frac{\sigma^2 s_i^2}{\Delta s^2} + \frac{r s_i}{\Delta s}\right)\Delta t,\\ p_d(s_i,t_j) &= \frac{1}{2}\left(\frac{\sigma^2 s_i^2}{\Delta s^2} - \frac{r s_i}{\Delta s}\right)\Delta t,\\ p_m(s_i,t_j) &= 1 - p_u - p_d. \end{aligned} \tag{33}$$

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 (17).

2.2 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. Lets summarise the high level differences:

- Risk-neutral probability measure Q is used for *pricing*, whereas *risks* problems are formulated under *real probability*, usually approximated by the historical probability.
- Prices are usually computed individually at trade level, whereas for *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.
- In *risks* we compute tails of a distribution, as quantiles; as opposed to the expectations used when pricing.

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,

$$\text{VaR}_\alpha = \inf\{l\in\mathbb{R} : \mathbb{P}(PL \le l) \ge \alpha\} = \inf\{l\in\mathbb{R} : F_{PL}(l) \ge \alpha\}, \tag{34}$$

where F_{PL} is the cumulative distribution function of the portfolio value variation random variable PL (as well known as profit and losses).

The VaR is therefore an estimate of how much one can gain or lose from one's portfolio over a given time horizon, with a given degree of confidence [91].

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

$$\text{CVaR}_\alpha = \frac{1}{1-\alpha} \int_\alpha^1 \text{VaR}_u du. \tag{35}$$

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

$$\text{CVaR}_\alpha = \mathbb{E}[PL | PL \geq \text{VaR}_\alpha], \tag{36}$$

or, in integral form,

$$\text{CVaR}_\alpha = \frac{1}{1-\alpha} \int_{\text{VaR}_\alpha}^{+\infty} x f_{PL}(x) dx, \tag{37}$$

where f_{PL} is the probability density function of PL. Thus, in the continuous case, the CVaR can be interpreted as the expected profit-loss in the event that VaR is exceeded.

In the following we will illustrate the dimensionality problem and some modelling alternatives with an example specific from Credit Risk. A complete and comprehensive monograph on risk management can be found in [64].

2.2.1 Credit Portfolio Management

A source of risk that needs to be measured and managed is the credit risk coming from the risk of default of a number of counterparties. 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 products or obligations (e.g. loans, bonds, overdrafts, etc) to a number of counterparties (also known as *obligors*). Each product is characterized by the *exposure at default* and the *loss given default*. The exposure at 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 its credit worthiness. Note that several products can be linked to the same counterparty, however here we will take the simplifying assumption that each counterparty has only one single product. 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 counterparties. 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 [65]. In this approach, a counterparty is assumed to go into default if the value of its assets falls 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 below 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 the credit quality of the counterparty 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 counterparty 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

$$D_j = \mathbb{1}_{\{V_j(T) < \tau_j\}} \sim Be\big[\mathbb{P}\big(V_j(T) < \tau_j\big)\big], \tag{38}$$

where $Be(p)$ is a Bernoulli distribution with probability of success p. Given D_j, the individual loss of counterparty j is defined as $L_j = D_j \cdot E_j$, while the total loss in the portfolio reads

$$L = \sum_{j=1}^{J} L_j. \tag{39}$$

As the credit quality of the counterparty (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 counterparties, and therefore credit portfolio models typically rely 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 introduced earlier (with $PL = L$).

2.2.2 Some Models for Portfolio Credit Risk

A common approach used to reduce the dimensionality of the problem is the introduction of factor models. In these models the credit quality of each counterparty is assumed to be driven by two different components: a set of factors that is shared by the counterparties and captures the systemic risk and a second that is unique to each counterparty and captures its idiosyncratic risk. Depending on the number of

factors of the systemic part, the model can be classified into the *one-* or *multi-factor* class. The power of factor models is that they allow us to reduce the number of parameters needed to capture the portfolio correlations. 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 the Merton model is defined as the logarithmic return of the asset value) of counterparty 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[6] 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:

$$X_j = \sqrt{\rho_j}\, Y + \sqrt{1-\rho_j}\, \varepsilon_j, \tag{40}$$

where Y and ε_j are i.i.d. standard normal random variables for all $j = 1, \dots, J$. Alternatively, as an extension of the model in Eq. (40), the *t-copula model* was introduced to take into account tail dependence,

$$X_j = \sqrt{\frac{\nu}{W}}\left(\sqrt{\rho_j}\, Y + \sqrt{1-\rho_j}\, \varepsilon_j\right), \tag{41}$$

where $\varepsilon_1, \dots, \varepsilon_J, Y \sim \mathcal{N}(0,1)$, 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 (40) 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 $\rho_1, \dots, \rho_J \in (0,1)$ are the correlation coefficients. In the case that $\rho_j = \rho$, for all $j = 1, \dots, J$, the parameter ρ is called the common asset correlation.

According to the Merton model described above, counterparty 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. counterparties in similar industrial sectors and geographies would typically be more correlated. For this, we consider the extension to multiple dimensions of the models presented in Sect. 2.2.2, 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 $[V_1, \dots, V_J]$ is determined by the multi-factor model,

$$X_j = \boldsymbol{a}_j^T \boldsymbol{Y} + b_j \varepsilon_j, \quad j = 1, \dots, J. \tag{42}$$

where $\boldsymbol{Y} = \left[Y_1, Y_2, \dots, Y_d\right]^T$ denotes the systematic risk factors. Note that we represent vectors by bold symbols. Here, $\boldsymbol{a_j} = \left[a_{j1}, a_{j2}, \dots, a_{jd}\right]^T$ 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,

$$\begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_J \end{bmatrix} = \begin{bmatrix} a_{11} \\ a_{21} \\ \vdots \\ a_{J1} \end{bmatrix} Y_1 + \begin{bmatrix} a_{12} \\ a_{22} \\ \vdots \\ a_{J2} \end{bmatrix} Y_2 + \cdots + \begin{bmatrix} a_{1d} \\ a_{2d} \\ \vdots \\ a_{Jd} \end{bmatrix} Y_d + \begin{bmatrix} b_1\varepsilon_1 \\ b_2\varepsilon_2 \\ \vdots \\ b_J\varepsilon_J \end{bmatrix}. \tag{43}$$

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

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

$$X_j = \sqrt{\frac{\nu}{W}}\left(\boldsymbol{a}_j^T \boldsymbol{Y} + b_j \varepsilon_j\right), \quad j = 1, \dots, J, \tag{44}$$

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

2.2.3 Numerical Methods for Risk Measures

Monte Carlo Assuming the loss distribution as defined in Eq. (39) and employing Monte Carlo methods, the VaR can be computed as follows:

- Generate M samples of the loss random variable, PL, denoted by $PL^1, PL^2, \dots, PL^M$.

[6] A copula is a mathematical object which allows to define the dependence structure between random variables. For more detail of the application of copulas in finance see [20]

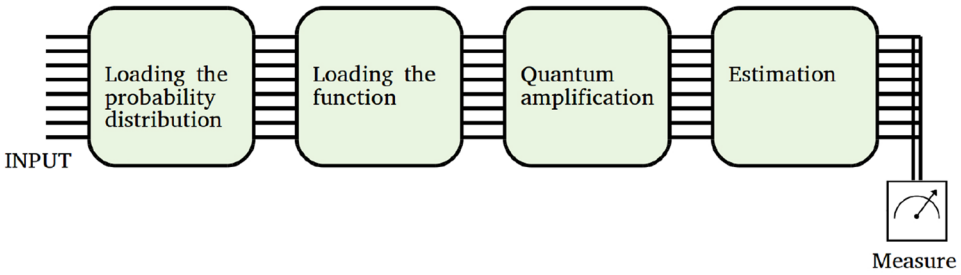

Fig. 2 Schematic pipeline of a quantum algorithm to compute the expected value of a function

- Compute the empirical CDF:[7]

$$\tilde{F}_{PL,M}(x) = \frac{1}{M}\sum_{m=1}^{M} \mathbb{1}_{\{PL^m \leq x\}}. \tag{45}$$

- Then, we have the estimator $\text{VaR}_\alpha \approx \tilde{F}^{-1}_{PL,M}(\alpha)$.

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

$$\text{VaR}_\alpha \approx \min_{PL^{\bar{m}}} \{\bar{m} \geq \lfloor \alpha M \rfloor\}, \tag{46}$$

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,

$$\text{CVaR}_\alpha \approx \frac{\sum_{m=1}^{M} \mathbb{1}_{\{PL^{\bar{m}} \geq \text{VaR}_\alpha\}} PL^{\bar{m}}}{\sum_{m=1}^{M} \mathbb{1}_{\{PL^{\bar{m}} \geq \text{VaR}_\alpha\}}}. \tag{47}$$

Notice that when sampling PL, 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 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, the utilization of *importance sampling* techniques being one of the most successful attempts.

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 Sect. 2.1.1) and in risk models (as the ones in Sect. 2.2.2), but also for stock value forecasting (usually in combination with other machine learning techniques).

As we have seen in Eq. (6), a multidimensional problem with N risk factors can be formulated through an $N \times N$ correlation matrix (ρ_{ij}), and an N volatility vector (σ_i). Equivalently, it can be formulated through the Hermitian and positive semi-definite covariance matrix Σ, the coefficients of which are $\Sigma_{ij} = \rho_{ij}\sigma_i\sigma_j$.

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

- 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.

3 Quantum Methods for Pricing and VaR

In Sect. 2 we discussed some of the most widely used numerical techniques to solve problems in pricing and VaR. Throughout this section we cover some proposals for their quantum counterparts.

In the first subsection, we cover Quantum-Accelerated Monte Carlo (QAMC), usually described in the literature as the quantum equivalents of Monte Carlo (see Sects. 2.1.3 and 2.2.3). In the second subsection, we cover techniques based on the PDE formulation (see Sect. 2.1.3). In the third subsection, we comment on some machine learning techniques (see Sects. 2.1.3 and 2.2.3). Finally, in the fourth subsection we discuss some of the challenges on the subject.

3.1 Quantum-Accelerated Monte Carlo

As we saw in Sects. 2.1.3 and 2.2.3, the computation of expectation values constitutes the basis for many pricing and risk problems. Such a computation is normally performed by means of classical Monte Carlo methods. The exponential growth of the dimension of the Hilbert space with the number of qubits, however, suggests that an alternative quantum implementation for Monte Carlo could be in a better position

[7] In (45) and (47) we use the indicator function of a set, $\mathbb{1}_C(x)$, which is equal to one if x belongs to C and equal to zero otherwise.

than its classical counterparts, especially over domains with a high dimensionality.

Quantum-Accelerated Monte Carlo[8] for the computation of the expected value of a function is usually divided in four steps (see Fig. 2):

1. **Loading the probability distribution** $p(x)$ into the quantum computer.
2. **Loading the function** $f(x)$ whose expected value we want to compute.
3. **Quantum amplification** of the expected value that we want to estimate. This part relies on Grover's amplification [41].
4. **Estimation** of the amplified expected value. This part can be performed in different ways: by means of an inverse quantum Fourier transform, through an approximate counting strategy or even with classical post-processing techniques.

In practice the functions $p(x)$ and $f(x)$ are appropriately discretized before they are loaded into the quantum computer. The discretization procedure is a relevant aspect of the computation which—in general—introduces approximations, this is however an aspect shared by classical and quantum algorithms. Besides, we are going to assume the existence of two oracles $\mathcal{P}$ and $\mathcal{R}$ which load the probability distribution $p(x)$ and the function $f(x)$, respectively. As we shall discuss in Sect. 3.4, this hypothesis might be too strong in some cases.

Under this considerations, QAMC algorithms have—at least—two very interesting properties:

- They are not based on any particularly restrictive assumption on the function whose expected value we want to compute, thereby encompassing quite generic cases. This is one of the reasons why they lie on a similar footing as classical Monte Carlo algorithms.
- They entail a theoretical quadratic speed-up with respect to classical Monte Carlo. More precisely, if the precision for classical Monte Carlo sampling generically scales as $M^{-\frac{1}{2}}$, where M is the number of samples, quantum algorithms have a theoretical precision scaling as M^{-1}.

In the following sections we start by giving a brief overview of the main amplification algorithms. Next, we describe what is *amplitude estimation* and discuss some algorithms to realize it. Then, we describe some concrete applications to the financial problem of pricing (see Sect. 2.1). Finally, we comment on the computation of risk indicators (see Sect. 2.2).

3.1.1 Amplitude Amplification

Given an oracle $\mathcal{A}$ defined as:

$$\mathcal{A}|0\rangle = a|\tilde{\Psi}_1\rangle + \sqrt{1-a^2}|\tilde{\Psi}_0\rangle\,, \tag{48}$$

the amplitude amplification algorithm gives us a strategy to enhance the probability of measuring $|\tilde{\Psi}_1\rangle$, namely the "good" or marked state. This strategy exploits similar ideas to the original Grover search algorithm.[9] The path to obtain the amplification consists in defining the Grover operator Q,

$$Q \equiv -\mathcal{A}S_0\mathcal{A}^{-1}S_f\,, \tag{49}$$

where the operator S_0 flips the sign of the component along $|0\rangle$ whereas the operator S_f flips the sign of $|\tilde{\Psi}_1\rangle$, that is to say, it flips the sign of the "good" states.

In order to understand the effect of the Grover operator, we rewrite Eq. (48) as

$$|\Psi\rangle = \sin(\theta)|\tilde{\Psi}_1\rangle + \cos(\theta)|\tilde{\Psi}_0\rangle\,, \tag{50}$$

where we have been able to associate an angle θ to the amplitude a because the state is properly normalised. An iterated application of Q leads to

$$\mathcal{Q}^k|\Psi\rangle = \sin\big((2k+1)\theta\big)|\tilde{\Psi}_1\rangle + \cos\big((2k+1)\theta\big)|\tilde{\Psi}_0\rangle\,, \tag{51}$$

where k denotes a generic integer. If one chooses k in (51) 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. Said otherwise, by measuring (51) we have a more efficient access to the estimation of a.

3.1.2 Amplitude Estimation

Once we have the amplified state (51), we still need to actually estimate a in the best possible way. We stress once more that (51) is the result of rotations—implemented by the Grover operator Q—aimed to enhance in a controlled way the probability to obtain $|\tilde{\Psi}_1\rangle$ upon measurement. So, a sampling by repeated preparations and measurements of (51), instead of the original vector $|\Psi\rangle$, already provides an advantage, although not quadratic. To guarantee a quadratic

[8] The paper [67] is generally regarded as representing the current state of the art in relation to quantum speedups in Monte Carlo tasks.

[9] The original Grover algorithm to search a marked element $\bar{x}$ corresponds to having $p(x) = 1/N$, which is the uniform sampling (implemented through a Walsh-Hadamard transform), while the oracle is $f(x) = \delta_{x\bar{x}}$, where δ is a Kronecker delta.

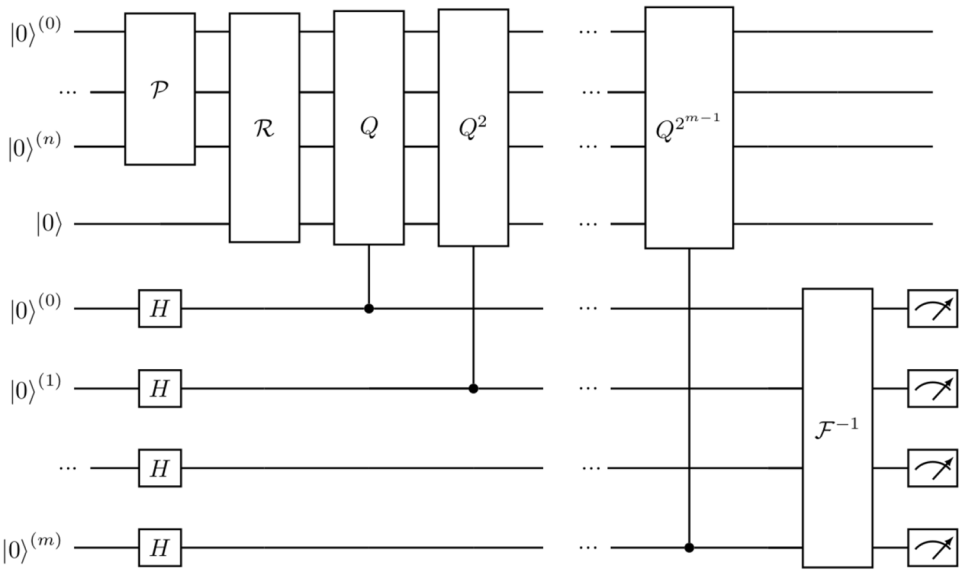

Fig. 3 Quantum circuit for amplitude amplification and estimation [15]

speed-up in the estimation of a by means of suitable theoretical bounds, we need a systematic way of leveraging the power of amplitude amplification. These systematic strategies are known in the literature as *amplitude estimation* techniques and in the following sections we describe some of them.

Original Amplitude Estimation The path suggested in the original paper [15] is to adopt an inverse quantum Fourier transform, see Fig. 3. In the picture, the oracle $\mathcal{A}$ of Sect. 3.1.1 corresponds to $\mathcal{R}(\mathcal{P} \otimes \mathbb{1})$ acting on the first n qubits. 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 constitute 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.

The inverse Quantum Fourier transform adopted by the original amplitude estimation algorithm [15] is resource demanding and thus constitutes a serious bottleneck, especially in relation to current or near-future technologies. Said otherwise, Quantum Fourier transform requires a deep and wide quantum circuit. For this reason, some algorithms which need less resources have been proposed.

Quantum Amplitude Amplification and Estimation with Max Likelihood One interesting possibility to avoid the inverse Quantum Fourier transform has been suggested in [84] 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 (51)). 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.

The classical post-processing adopts standard statistical tools such as Fisher information and the Cramer–Rao inequality, we refer to [22] for their description. The Max likelihood alternative for amplitude estimation has been further discussed in subsequent papers. In [39] the authors stress that the accuracy of the Max likelihood method [84] 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 [1].

Iterative Quantum Amplitude Estimation In [39] 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

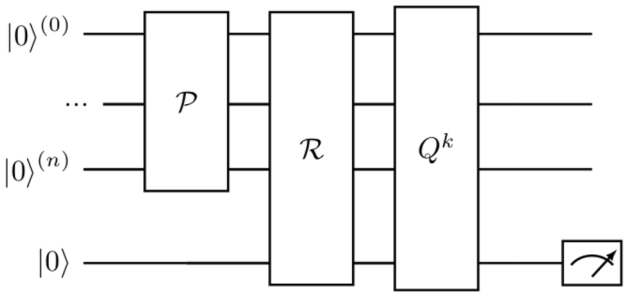

Fig. 4 Quantum circuit for amplitude amplification and estimation with max likelihood [84]. The circuit depicted refers to a specified value of the amplification exponent k. A collection of similar circuits for all the desired values of k is needed

is able to reduce the number of qubits and gates, making the overall algorithm lighter. The analysis in [39] focuses on a rigorous study of the quadratic quantum speed-up for their algorithm.

To achieve the quadratic speed-up, the iterative quantum amplitude estimation algorithm—like the other amplitude estimation algorithms—loads the square root of the integrand function in the quantum state. Instead, an alternative method dubbed quantum coin (to be covered in Sect. 3) loads the function to the quantum state directly, rather than its square root (something which is sometimes referred to as *direct embedding* [57] or amplitude encoding). Despite of the different embedding, 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 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 sin function in (51) is invertible. Thus, one can obtain a finer estimation of θ, and thereby of a too.

Power-Law and QoPrime Amplitude Estimation Both Power-law and QoPrime algorithms have been described in [34]. The paper frames the asymptotic trade-off between the quantum speed-up of an amplitude estimation algorithm and its depth. More precisely, the authors relate the speed-up 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 [84]. 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 [84]. 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.

Quantum Coins The "quantum coin" [1, 3, 81] also offers a way to circumvent the use of the inverse Quantum Fourier Transform 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.

3.1.3 Applications to Option Pricing

As presented in Eq. (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:[10]

$$\mathbb{E}[\hat{f}(\hat{x})] = \int_{\Omega} \hat{p}(\hat{x})\hat{f}(\hat{x}) \,, \tag{52}$$

where the domain Ω can be multi-dimensional. In pure computational terms, the problem can be reduced to that of computing an expression like:

$$\sum_{x \in X} p(x) f(x) \approx \int_{\Omega} \hat{p}(\hat{x})\hat{f}(\hat{x}) = \mathbb{E}[\hat{f}(\hat{x})] \,, \tag{53}$$

where

$$\sum_{x \in X} p(x) = 1 \,. \tag{54}$$

Here we have implicitly defined p, f and X, which are proper discretized versions of $\hat{p}$, $\hat{f}$ and Ω respectively. Recall that, upon the discretization procedure (which we are not specifying to keep the treatment general), the sum in (53) corresponds to the discretized version of the original integral we needed to compute. The problem of defining a suitable (and

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

efficient) discretization procedure, common to classical and quantum algorithms, entails the theory of measure and the choice of optimal meshes.

Up to this point, there is no difference between the classical and the quantum approaches. To be able to continue the process of computing (53) in the quantum computer, 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 and then rescale back at the end of the computation, exploiting the linearity of expected values. Hence, there is no loss of generality in the assumption[11]

$$f : x \in X = \{0,1\}^n \to [0,1] \,. \tag{55}$$

With f and p defined accordingly to the restrictions present in a quantum computer, we can continue by assuming 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

$$\mathcal{P}|0\rangle_n = \sum_{x=0}^{2^n-1} \sqrt{p(x)}\, |x\rangle_n \,, \tag{56}$$

and

$$\mathcal{R}|x\rangle_n|0\rangle = |x\rangle_n \Big(\sqrt{f(x)}|1\rangle + \sqrt{1-f(x)}|0\rangle \Big) \,. \tag{57}$$

We have indicated with $|x\rangle_n$ a quantum register composed of n qubits which has an affine mapping with x. In (57) 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 (57) and (56), we obtain

$$|\Psi\rangle \equiv \mathcal{R}(\mathcal{P}\otimes \mathbb{1})|0\rangle_n|0\rangle = \sum_{x=0}^{2^n-1} \sqrt{p(x)}\, |x\rangle_n \Big(\sqrt{f(x)}|1\rangle\sqrt{1-f(x)}|0\rangle \Big), \tag{58}$$

which corresponds to the oracle $\mathcal{A} \equiv \mathcal{R}(\mathcal{P}\otimes\mathbb{1})$ of Sect. 3.1.1. The probability of getting the auxiliary qubit equal to 1 in a measurement of the state $|\Psi\rangle$ is given by

$$\mathbb{E}_{x\sim p}(f) = \sum_{x=0}^{2^n-1} p(x) f(x) \,, \tag{59}$$

which is actually the expectation value we want to compute.

We have almost mapped our original problem (53) to that of suitably measuring (58). To clarify the mapping, one defines the following vectors:

$$\begin{aligned} |\Psi_1\rangle &= \sum_{x=0}^{2^n-1} \sqrt{p(x)f(x)}\, |x\rangle_n|1\rangle \,, \\ |\Psi_0\rangle &= \sum_{x=0}^{2^n-1} \sqrt{p(x)(1-f(x))}\, |x\rangle_n|0\rangle \,. \end{aligned} \tag{60}$$

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. Notice that, from the definitions in (60), we have

$$|\Psi\rangle = |\Psi_1\rangle + |\Psi_0\rangle \,. \tag{61}$$

One can also define

$$a^2 \equiv \langle\Psi|\Psi_1\rangle = \langle\Psi_1|\Psi_1\rangle = \sum_{x=0}^{2^n-1} p(x) f(x) \,, \tag{62}$$

$$1-a^2 \equiv \langle\Psi|\Psi_0\rangle = \langle\Psi_0|\Psi_0\rangle = \sum_{x=0}^{2^n-1} p(x)(1-f(x)) \,, \tag{63}$$

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 (60), namely

$$\begin{aligned} |\tilde{\Psi}_1\rangle &= \frac{1}{a} \sum_{x=0}^{2^n-1} \sqrt{p(x)f(x)}\, |x\rangle_n|1\rangle \,, \\ |\tilde{\Psi}_0\rangle &= \frac{1}{\sqrt{1-a^2}} \sum_{x=0}^{2^n-1} \sqrt{p(x)(1-f(x))}\, |x\rangle_n|0\rangle \,. \end{aligned} \tag{64}$$

Thus, we have

$$|\Psi\rangle = a\, |\tilde{\Psi}_1\rangle + \sqrt{1-a^2}\, |\tilde{\Psi}_0\rangle \,. \tag{65}$$

From this point on, the amplitude amplification and estimation algorithms described in Sect. 3.1.2 take over.

The simplest use-cases in finance for this procedure are pricing problems for vanilla European options (see Sect. 2.1.1.1, for details). They are considered in the literature as a first benchmark, or better a proof-of-concept, for quantum implementations, see for example [53, 71, 73, 83].

3.1.4 Applications to Risk Analysis

As described in Sect. 2.2, some of the tools used for option pricing can be leveraged for financial risk analysis. In particular, both in the pricing and in the risk assessment arena, Monte Carlo methods (see Sects. 2.1.3 and 2.2.3)

[11] It is however important to recall that such rescaling affects the final estimation of errors.

play a predominant role. Nevertheless, risk analysis problems require a precise estimation of the tail of a distribution, which constitute a *more* demanding task in terms of Monte Carlo samplings. On top of that, the problem usually is highly dimensional, because portfolios of various derivatives are usually considered.

Classical strategies to improve the performance of Monte Carlo resort to importance sampling, but, also after such mitigation, the problem remains usually very heavy in terms of the necessary 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 Sect. 3.1.3, such a discontinuous activation can be implemented by means of a *quantum comparator circuit*, see [24] for details.

In risk assessment, one can be interested in estimating the probability of experiencing a future loss exceeding a predetermined value, according to, for example, the definition of the VaR and the CVaR risk measures as in Sect. 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 asking which maximal loss corresponds to it. This question can be addressed combining a comparator with a binary search algorithm [26, 27].

As a final remark on quantum-enhanced Monte Carlo techniques, we refer to [7] for the quantum generalization of the classical multi-level Monte Carlo strategy [32]. 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 [7] is to apply a quantum circuit to solve each Monte Carlo sub-problem.

3.2 Quantum PDEs

Following [36], the quantum approaches rely on the observation that the financial PDEs can be mapped into the propagation governed by an appropriate Hamiltonian operator.

To this purpose, the first step is to consider appropriate changes of variable and/or unknown in the Black–Scholes equation (10). 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 $S = e^x$, Eq. (10) becomes

$$\frac{\partial V}{\partial t} + \left(\mu - \frac{\sigma^2}{2}\right)\frac{\partial V}{\partial x} + \frac{\sigma^2}{2}\frac{\partial^2 V}{\partial x^2} - \mu\, V = 0\,, \tag{66}$$

which can be written as

$$\frac{\partial V}{\partial t} = -i\hat{H}_{\mathrm{BS}}\, V\,, \tag{67}$$

where

$$\hat{H}_{\mathrm{BS}} = i\frac{\sigma^2}{2}\hat{p}^2 - \left(\frac{\sigma^2}{2} - \mu\right)\hat{p} + i\mu\mathbb{1}\,, \tag{68}$$

and we have defined the momentum operator

$$\hat{p} = -i\frac{\partial}{\partial x}\,. \tag{69}$$

Note that (67) is a Schrödinger-like equation. However, it is important to stress that the Hamiltonian $\hat{H}_{\mathrm{BS}}$ defined in (68) is *not* Hermitian. Therefore, the associated evolution operator

$$\hat{U}(t, t_0) = e^{-i\hat{H}_{\mathrm{BS}}(t-t_0)}\,, \tag{70}$$

is not unitary. For implementing the evolution operator (70) into a quantum circuit, i.e. through unitary operations, one can consider an enlarged system.[12] In [36], $\hat{U}(t, t_0)$ 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 [30]. Instead of doubling the system, 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 [87] for details) so that (66) maps into the heat equation

$$\frac{\partial v}{\partial \tau} = \frac{1}{2}\frac{\partial^2 v}{\partial x^2}\,. \tag{71}$$

Next, using the Wick rotation $\tilde{\tau} = -i\tau$, which maps real time to imaginary time, the heat equation (71) turns into a Schrödinger-like equation, namely

$$\frac{\partial v}{\partial \tilde{\tau}} = -\hat{H}_{\mathrm{HE}}\, v\,, \tag{72}$$

with

[12] 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 wants to consider dissipation.

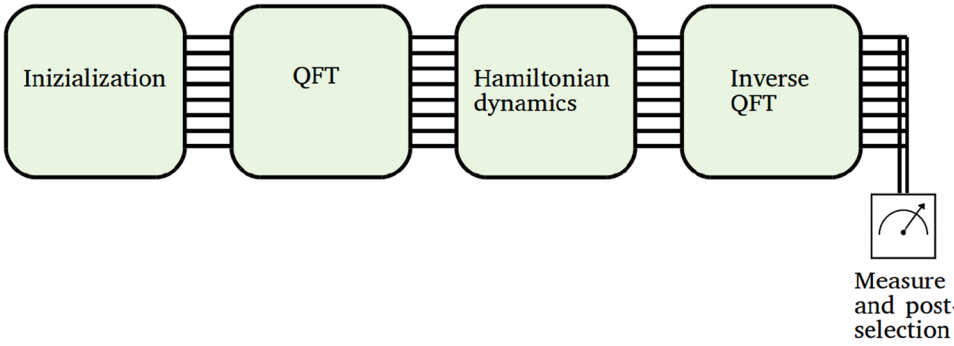

Fig. 5 Schematic pipeline of the quantum algorithm described in [36]. The acronym QFT stands for Quantum Fourier Transform

$$\hat{H}_{\mathrm{HE}} = -\frac{i}{2}\hat{q}^2\,, \quad \hat{q} = -i\frac{\partial}{\partial x}. \tag{73}$$

This leads to a purely anti-Hermitian Hamiltonian operator. Said otherwise, (72) 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 (72) can be associated to a non-unitary (read dissipative) cooling evolution. These observations are relevant in practice, especially because they allow us to connect to an area where similar problems have been thoroughly investigated, namely that of finding the ground state of quantum systems. 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.

3.2.1 Propagating with $\hat{H}_{\mathrm{BS}}$

In [36] they consider the Hamiltonian (68) 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 payoff function. These issues remain as open problems. A schematic depiction of the algorithm is given in Fig. 5.

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. 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 [36] 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 advantage 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.

3.2.2 Imaginary-Time Propagation with $\hat{H}_{\mathrm{HE}}$

Imaginary-time propagation is a standard technique in physics and it is related to mapping the real time t of an evolution to imaginary time $\tau \equiv it$, an operation that, depending on context, is sometimes referred to as Wick rotation. We do not enter into the details and implications of imaginary-time propagation in general, yet intuitively it can be thought as a technical device which allows to exploit better convergence properties, for instance trading oscillatory behaviours with damped ones.

A quantum algorithm for imaginary-time propagation has been developed in the field of quantum chemistry [63].[13] It assumes to deal with a Hamiltonian

$$\hat{H} = \sum_i \lambda_i \hat{h}_i\,, \tag{74}$$

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 [30] 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 [63] they

[13] For related discussions we refer to [85] and [47].

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 that of trading 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

$$\delta \left\| \left(\frac{\partial}{\partial \tau} + \hat{H} + E \right) |\psi(\Theta(\tau))\rangle \right\| = 0 \,, \tag{75}$$

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 [63] for details.

The evolution of the parameters is derived from the linear system of differential equations associated to the variational principle (75), namely

$$\sum_j A_{ij} \dot{\theta}_j = b_i \,, \tag{76}$$

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

$$A_{ij} = \mathrm{Re}\left[\frac{\partial \langle \psi|}{\partial \theta_i} \frac{|\partial \psi\rangle}{\partial \theta_j} \right] , \quad b_i = -\mathrm{Re}\left[\sum_a \lambda_a \frac{\partial \langle \psi|}{\partial \theta_i} \hat{h}_a |\psi\rangle \right] . \tag{77}$$

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, (71), occurs for the Black-Scholes model but is expected not to hold when generalizing it. Secondly, the evolution of the parameters is governed by (76) 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 (76) still within the quantum circuit, possibly implementing the Harrow–Hassidim–Lloyd (HHL) algorithm [43] or its refinements (see for example [18]). For further discussions about solving partial differential equations in quantum computers we refer to [58].

3.3 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) front of research 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 [12]. Many 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 summarise 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 2. Note however that most quantum machine learning 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 [14]. For a discussion about the relation among quantum machine learning and kernel methods we refer to [77]. Note as well that some (not all) of the QML algorithms can be *dequantized*, i.e., transformed into a classical quantum-inspired equivalent (see [33, 86], for example). Although useful to study theoretical questions about possible quantum speed-ups and a fair comparison among classical and quantum algorithms, dequantization might not be particularly helpful to the purpose of designing actual classical algorithms.

3.3.1 Quantum Principal Component Analysis

In Sect. 2.2.3 we have briefly described the usage of the PCA algorithm in finance. The first step behind PCA is to calculate the eigenvalues of the covariance matrix $\Sigma \in \mathbb{R}^N \times \mathbb{R}^N$, which is Hermitian and positive semi-definite. To this purpose, one possibility is to use the Quantum Phase Estimation (QPE) algorithm. However, two problems arise: (i) an initial state describing all the eigenvectors of the covariance matrix has to be loaded; (ii) the covariance matrix must be decomposed as a summation of Pauli strings (which needs N^2 classical operations). Taking into account that Σ is usually a dense matrix (i.e. not sparse), there is no guarantee of obtaining a significant speed-up, as that achieved by other algorithms like HHL.

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 [60]. It describes an algorithm to obtain the exponential of a density matrix using C copies of it. One concrete example of this technique was implemented by [2] in the case $N = 2$, with $C = 2$. Having enough resources, QPCA theoretically runs exponentially faster. Other refined versions have been proposed later [44, 59].

Recently, [92] suggested to calculate the eigenvalues and eigenvectors of Σ through a variational algorithm, by using the density matrix expansion

$$\rho = \sum_{j=0}^{N-1} \lambda_j |\psi_j\rangle\langle\psi_j| \,, \tag{78}$$

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

$$\rho_f = U(\Theta)\rho\, U(\Theta)^\dagger = \sum_{j=0}^{N-1} \lambda_j |j\rangle\langle j| \,, \tag{79}$$

where $\{|j\rangle\}$ is the usual computational basis and $\Theta = \{\theta_i\}$ is the set of angles that define the operator $U(\Theta)$. This unitary operator is searched for by optimizing the parameters Θ using variational hybrid algorithms. Once $U(\Theta)$ is found (or—at least—suitably approximated), the eigenvalues can be computed directly by measuring probabilities on the computational basis. It is nevertheless not clear yet whether this procedure is efficient in the general case.

The workflow to use these density-matrix-based QPCA algorithms should include several steps [2]:

- Convert Σ in a density matrix ρ. For this purpose, three characteristics of the density matrix must be fulfilled: it must be Hermitian, positive semi-definite and its trace must be equal to 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.
- In general, ρ represents a mixed state. As quantum computers can only work with pure states, ρ must be purified. This means that, in order to load ρ into a quantum circuit, $n = 2\log(N)$ qubits and additional classical preprocessing are required.
- The purified state must be loaded into the quantum circuit, which could need a large number of gates (see Sect. 3.4.1 for a discussion about the loading problem).

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

3.3.2 Regression

Classical and more advanced (neural networks-based) regression methodologies are greatly appreciated in derivative pricing problems, in particular for options with early exercise, like American options (see Sect. 2.1.1), 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 [43] is a first relevant representative algorithm 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 [90]. Some related works in this field are [78, 89]. A more involved approach is presented in [74], 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* [79] or *Quantum Neural Networks* [10] (and the references therein). Other quantum-based techniques like *Quantum Kernel Estimation* [28] and QML with Gaussian processes [94] have been recently proposed.

3.3.3 Quantum Neural Networks and Deep Learning

The drastic increase in the computational power has enabled the use of deep Artificial Neural Networks (ANN) for general purpose, giving rise to the so-called *deep learning* techniques. This computational paradigm has boosted the applicability of approaches based on 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 Sect. 2.1.3, we have presented a PDE-based problem formulation suitable to be solved by unsupervised ANN approaches.

In the context of deep learning and ANNs, 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 computational efficiency in performing the training and/or avoiding possible undesirable malfunctioning (local minima) with respect to the classical alternatives (e.g. stochastic gradient descent). The advances on training the so-called *Boltzmann machines* (particularly the Restricted Boltzmann Machines (RBMs))

are especially relevant. Some representative works in this area of interest are [4, 5, 51, 88].

Another research line consists in 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 [10, 23]. 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 Sect. 3.4.1).

3.4 Discussion

In this section we discuss some of the open problems in quantum computing for option pricing and VaR. As QAMC approaches to solve pricing and VaR problems are the most widely adopted ones in the literature and the ones taking more attention, we focus on them. The main implicit assumption in QAMC is the existence of an efficient oracle which loads the probability distribution. Both for pricing and VaR, loading the distribution means that it is necessary to create a circuit for a unitary operation $\mathcal{P}$ such that:

$$|P\rangle = \mathcal{P}|0\rangle = \sum_{i=0}^{N-1} \sqrt{P_i}|i\rangle\,, \quad \sum_{i=0}^{N-1} P_i = 1\,. \tag{80}$$

In the case of VaR, the cost of creating such a unitary can be mitigated to some extent using the techniques from Sect. 3.4.1. In the case of pricing, it is much more critical as we discuss below.

In pricing, the distribution to be loaded has to be previously generated through the simulation of a SDE. As it was discussed in the first part of this survey, the simulation of the SDE consumes most of the computing resources. When assessing the overall performance of the QAMC one must take into account this step. Otherwise, the latter comparison of the QAMC and the CMC would be unfair. Indeed, the claims of a quadratic speed-up of the QAMC over the CMC for financial applications—in general—do not take into account the generation step. If we compare both QAMC and CMC under the same conditions, with the approaches proposed in the literature, we will find that there is no rigorous evidence for the quadratic speed-up.

If we assume that we have the probability distribution in the classical case (as it is done for the quantum one), the problem of pricing is reduced to computing the following expectation

$$\mathbb{E}[f] = \sum_{i=0}^{N-1} p(x_i) f(x_i)\,, \tag{81}$$

where p is the probability distribution, f de payoff function and x_i are the points where we know the probability distribution. In this case the number of operations performed in the classical computer is of order N and there is no quadratic speedup for the QAMC. In fact, when adding the costs of loading the probability distribution and the payoff into the quantum state we might end up with a clear disadvantage.

These problems provide concrete examples about possible issues encountered in designing *full* quantum algorithms able to reach a quantum advantage. Almost any speed-up concentrated in a subroutine of an overarching inefficient algorithm, however interesting, is clearly 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 algorithms studied in the literature are not viable on current or near-future quantum technology[14]. 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 [8], where it is argued that a quadratic speed-up is not sufficient to obtain a quantum advantage, mainly due to the—constant but large—resource overheads (needed for error correction). An important overarching suggestion emerging from [8] is that the complexity scaling is in general not enough to properly define an actual threshold for quantum advantage.

In [19] 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 10 MHz (current quantum technologies moves on the order of 10 kHz). 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

[14] It is difficult to forecast the evolution of the quantum computing technology in the next 5 to 10 years. We refer here to have only a few dozens of qubits (physical or logical) with errors in operations higher than 10^{-5} which can execute circuits with only a small number of steps—around 1000—before the result collapses to useless or meaningless values

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.

In the following subsections we make some further comments on:

1. Loading a probability distribution: the direct benefits in this matter goes to VaR problems.
2. Generating a probability distribution: this would be the direct equivalent of simulating a stochastic differential equation.
3. Loading a payoff function: this case is only relevant for pricing.

3.4.1 Specific Methods to Load Probability Distributions

Loading the necessary state into a quantum register in order to initialize an algorithm constitutes often the main bottleneck. For example, [31, 67] show that in QAMC the overall calculation time is dominated by the (asymptotic) time to load the probability distribution as

$$T_{\text{QAMC}} = O\left(\frac{T_{\mathcal{P}}}{\epsilon}\right), \tag{82}$$

where T_{QAMC} is the time cost of the ideal amplitude estimation algorithm, $T_{\mathcal{P}}$ the time for implementing the oracle $\mathcal{P}$ which loads the distribution and ϵ the desired sampling precision.

As opposed to pricing where there are cases in which the probability distribution can be generated sinthetically, in VaR we always need to load an empirical distribution. Thus, the loading step is of particular importance for VaR.

If we were to load the exact[15] probability distribution into the quantum state, we could leverage general techniques for loading functions, such as:

- Use a general method to convert unitary operators to circuits [38].
- Use specific methods to initilize the amplitudes to a normalized vector [54, 80, 82].
- Grover approach: use the properties of the probability distribution to create an efficient circuit [40].

The main problem of the first two methods is the poor scalability with the number of points that we need to load. In order to seek for efficient loading algorithms, we can give up techniques which load the exact function in the quantum state and explore the possibility of approximating the state to be loaded. In this way we lower the resources necessary to load the state at the cost of accuracy and exploit the fact that we are working with a specific subset of functions. This generic idea of loading an approximated version of the state, aiming to have an efficient process, can take different practical paths. We focus on two of them:

- Variational Quantum Circuits: create an *ad-hoc* circuit which approximates the amplitudes [68].
- Tensor Networks techniques [31, 48, 72].

Grover Approach [40] proposed a general methodology to load integrable probability distributions into the states efficiently. The basic idea is to discretize the probability density $p(x)$, defined on a continuous one-dimensional space spanned by x, in 2^n regions iteratively, splitting in each step one region in two. In the first step, the initial state is prepared as:

$$|0\rangle \otimes |0\rangle_{n-1} \rightarrow (\sqrt{p_l}|0\rangle + \sqrt{p_r}|1\rangle) \otimes |0\rangle_{n-1}\,, \tag{83}$$

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

$$\theta_i = \arccos\left(\sqrt{f(i)}\right), \qquad \text{with} \qquad f(i) = \frac{\int_{x_L^i}^{\frac{x_R^i + x_L^i}{2}} p(x)dx}{\int_{x_L^i}^{x_R^i} p(x)dx}\,, \tag{84}$$

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:

[15] With *exact* we mean that the only approximation is—possibly—due to the discretization.

$$\sum_{i=0}^{2^m-1} \sqrt{p_i^{(m)}}|i\rangle_m|0\rangle_{(n-m)}|0\rangle_q \tag{85}$$

$$\begin{aligned}
&\longrightarrow \sum_{i=0}^{2^m-1} \sqrt{p_i^{(m)}}|i\rangle_m|0\rangle_{(n-m)}|\theta_i\rangle \\
&\longrightarrow \sum_{i=0}^{2^m-1} \sqrt{p_i^{(m)}}|i\rangle_m\Big(\cos(\theta_i)|0\rangle + \sin(\theta_i)|1\rangle\Big)|0\rangle_{(n-m-1)}|\theta_i\rangle \\
&\longrightarrow \sum_{i=0}^{2^m-1} \sqrt{p_i^{(m)}}|i\rangle_m\Big(\cos(\theta_i)|0\rangle + \sin(\theta_i)|1\rangle\Big)|0\rangle_{(n-m-1)}|0\rangle_q \\
&\longrightarrow \sum_{j=0}^{2^{m+1}-1} \sqrt{p_j^{(m+1)}}|j\rangle_{m+1}|0\rangle_{(n-m-1)}|0\rangle_q \,.
\end{aligned} \tag{86}$$

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 (85) to (86), 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, some recent works dispute the possibility of a speed-up when this method is used for Monte Carlo [19, 45, 52]. Similar algorithms which use the conditioned probabilities are presented in [54].

It is important to stress that all these methods based on the Grover approach (or variations thereof) rely on the knowledge of an analytic expression for the PDF, in general, with some further assumptions such as log-concavity. When dealing with empirical data, like in VaR calculations, the computation of the conditioned probabilities (84) is straightforward and can be implemented by means of a simple circuit with controlled R_y gates using binary trees [44, 55].

Variational Circuits With Variational Circuits we loosely refer to all methods which rely on a parametric ansatz whose parameters are chosen by means of some optimization procedure [11]. 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 want to pay particular attention to a specific strategy to train the circuit called Quantum Generative Adversarial Networks (QGANs).

QGANs have been proposed by [75, 95]. They are based on the classical concept proposed by [37], where two deep learning models are trained simultaneously in an adversarial setup. One model G, called the generator, tries to generate data that are checked by a second model D, the discriminator. This latter tries to distinguish if the data comes from a real distribution or not. For the quantum case, G is also a PQC which is trained to decieve the discriminator, which can be a classical deep learning model. Once the generator is trained, it can be used alone to load the desired distribution. The discriminator could be a quantum circuit or a classical deep learning model, while G is a parametric circuit $G(\Theta)$ that transforms an initial PDF into the desired PDF. The initial distribution can be a uniform, a normal, or a generic random distribution. Thus, we have

$$|P\rangle = G(\Theta)U|0\rangle \,, \tag{87}$$

where U loads the initial distribution and $G(\Theta)$ transforms it to the desired one. In the case that the initial distribution is the uniform distribution, U 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. In [95] the authors used QGANs to make an experiment to solve the option pricing problem. In their experiment, an initial normal PDF has proved the best to obtain a desired PDF according to the Kolmogorov-Smirnov distance. In the case of [75], the circuit has a different mechanism to take into account the latent space (i.e. the initial PDF). 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 state is loaded as

$$|P\rangle = G(\Theta)U(f(z))|0\rangle \,. \tag{88}$$

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 [93].

In general, variational methods do not provide a clear path to get the sought for speed-up. Specifically, it is difficult to guarantee the loading performance by means of rigorous bounds. This is mainly due to the uncertainty associated to the optimization of the parameters [19]. For each data distribution one needs to train the circuit, which clearly requires time and consumes resources. A possible mitigation comes from the fact that after having trained an initial circuit for an asset, it is sometime possible to assume that the associated distribution changes slowly so that just an daily and automatic small retraining would be needed daily.

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[16]. 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 the 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 [72].

Matrix product states (MPS) constitute a widely used class of tensor network ansatzes. These have been long studied in the context of quantum computation [76] 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 [48].

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 [25] and generic multivariate analysis [31] (e.g. expected value, Fourier transform, interpolation and solving PDE). In fact, in [31] the author presents an algorithm to efficiently use MPS techniques to encode smooth, differentiable multivariate functions in quantum registers. Although being a promising method, the extension to a general empirical data-based probability is not, to our knowledge, yet known and needs further research.

3.4.2 Specific Methods to Generate Probability Distributions

With "generative approach" we refer to the actual simulation of the discrete stochastic processes which generate the desired approximate 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 it usually happens in pricing.

In the class of generative techniques we find the variational quantum simulation [29], implemented—for example—by means of trinomial trees [57] (see Sect. 2.1.3)[17]. In [57] the up/down transition probabilities (32) 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 [57] relies on the linear combination of unitaries [21]. In [17] 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 [19].

Another interesting member of the class of generative techniques is given by quantum walks [6, 7, 62, 67]. These can be seen 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 [16]. To clarify the ideas behind quantum walks, 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 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 [69]. 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"[18] 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

[16] In the tensor network literature, the auxiliary space is usually referred to as *bond dimension* or *virtual dimension*. It refers to the dimensionality of the index that connects tansors. A higher-dimensional auxiliary space corresponds to a richer (and more complex) ansatz which is correspondingly more expressive as well as more costly in terms of resources.

[17] [42] contains a study on the quantum implementation of the binomial tree approach. Note that we can see the *n*-nomial approach as a way to address the quantum numerical solution of a partial differential equation [30, 31, 36] (see Sect. 3.2).

[18] Not to be confused with the quantum coin algorithm we have introduced for integration.

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.

3.4.3 Specific Methods to Load the Payoff Function

The task of computing the expectation value of a payoff function with respect to a probability distribution, see Eq. (53), requires that both be loaded into the quantum circuit, accordingly to the pipeline depicted in Fig. 2. As already commented below Eq. (53), the probability distribution and the payoff function are in general loaded separately. Nevertheless, they can be loaded simultaneously [17].

The payoff functions for European vanilla options, see Eq. (1), are relatively easy to implement by means of a comparator circuit. 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[19].

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–quite generically—needs to load the payoff on a quantum state which already encodes the probability distribution. On the contrary, one typically 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 pointwise. However this is clearly an approach which scales inefficiently and cannot be planned for real applications.

4 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 is 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 allow us to circumvent the use of resource-demanding Quantum Fourier Transforms, providing grounds for optimism.

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 payoff 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 [49], 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 to 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.

Acknowledgements All authors acknowledge the European Project NExt ApplicationS of Quantum Computing (NEASQC), funded by Horizon 2020 Program inside the call H2020-FETFLAG-2020-01 (Grant Agreement 951821). Á. Leitao, A. Manzano and C. Vázquez wish to acknowledge the support received from the Centro de Investigación de Galicia "CITIC", funded by Xunta de Galicia and the European Union (European Regional Development Fund- Galicia 2014-2020 Program), by Grant ED431G 2019/01.

[19] See [24] for the construction of a quantum comparator circuit as a modification of a ripple adder.

References

1. Aaronson S, Rall P (2020) Quantum approximate counting, simplified. In Proceedings of Symposium on Simplicity in Algorithms, SIAM, pp 24–32
2. Adedoyin A, Ambrosiano J, Anisimov P, Bärtschi A, Casper W, Chennupati G, Coffrin C, Djidjev H, Gunter D, Karra S, Lemons N (2020) Quantum algorithm implementations for beginners. arXiv:1804.03719
3. Abrams DA, Williams CP (2004) Fast quantum algorithms for numerical integrals and stochastic processes. arXiv:9908083
4. Adachi SH, Henderson MP (2015) Application of quantum annealing to training of deep neural networks. arXiv:1510.06356
5. Alcazar J, Leyton-Ortega V, Perdomo-Ortiz A (2020) Classical versus quantum models in machine learning: insights from a finance application. Mach Learn 1(3):035003
6. Ambainis A, Kempe J, Rivosh A (2005) Coins make quantum walks faster. In: Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '05, SA. Society for Industrial and Applied Mathematics pp 1099–1108
7. An D, Linden N, Liu JP, Montanaro A, Shao C, Wang J (2020) Quantum-accelerated multilevel Monte Carlo methods for stochastic differential equations in mathematical finance. arXiv:2012.06283
8. Babbush R, McClean JR, Newman M, Gidney C, Boixo S, Neven H (2021) Focus beyond quadratic speedups for error-corrected quantum advantage. PRX Quantum 2:010103
9. C. Beck, M. Hutzenthaler, A. Jentzen, and B. Kuckuck. An overview on deep learning-based approximation methods for partial differential equations, 2021. arXiv:2012.12348
10. K. Beer, D. Bondarenko, T. Farrelly, T. J. Osborne, R. Salzmann, D. Scheiermann, and R. Wolf. Training deep quantum neural networks. *Nature Communications*, 11(808), 2020
11. Benedetti M, Lloyd E, Sack S, Fiorentini M (2020) Parameterized quantum circuits as machine learning models. Quantum Science and Technology 5(1):019601
12. Biamonte J, Wittek P, Pancotti N, Rebentrost P, Wiebe N, Lloyd S (2017) Quantum machine learning. Nature 549:195–202
13. Black F, Scholes M (1973) The pricing of options and corporate liabilities. Journal of Political Economy 81(3):637–654
14. A. Bouland, W. van Dam, H. Joorati, I. Kerenidis, and A. Prakash. Prospects and challenges of quantum finance, 2020. arXiv:2011.06492
15. Brassard G, Hoyer P, Mosca M, Tapp A (2000) Quantum amplitude amplification and estimation. AMS Contemporary Mathematics Series 305:06
16. M. C. Braun, T. Decker, N. Hegemann, S. F. Kerstan, and C. Schäfer. A quantum algorithm for the sensitivity analysis of business risks, 2021. arXiv:2103.05475
17. A. Carrera Vazquez and S. Woerner. Efficient state preparation for quantum amplitude estimation. *Physical Review Applied*, 15:034027, Mar 2021
18. A. Carrera Vazquez, R. Hiptmair, and S. Woerner. Enhancing the quantum linear systems algorithm using Richardson extrapolation, 2020. arXiv:2009.04484
19. S. Chakrabarti, R. Krishnakumar, G. Mazzola, N. Stamatopoulos, S. Woerner, and W. J. Zeng. A threshold for quantum advantage in derivative pricing, 2020. arXiv:2012.03819
20. U. Cherubini, E. Luciano, and W. Vecchiato. *Copula methods in finance*. John Wiley & Sons, 2004
21. Childs AM, Wiebe N (2012) Hamiltonian simulation using linear combinations of unitary operations. Quantum Information & Computation 12(11–12):901–924
22. T. M. Cover. *Elements of information theory*. Wiley, Hoboken, N.J., 2nd ed. edition, 2005
23. Coyle B, Henderson M, Le JCJ, Kumar N, Paini M, Kashefi E (2021) Quantum versus classical generative modelling in finance. Quantum Science and Technology 6(2):024013
24. Cuccaro SA, Draper TG, Kutin SA, Petrie Moulton D (2004) A new quantum ripple-carry addition circuit. arXiv:0410184
25. Dang A, Hill CD, Hollenberg LCL (2019) Optimising matrix product state simulations of Shor's algorithm. Quantum 3:116
26. Egger DJ, Woerner S (2019) Quantum risk analysis. Quantum Inf 5(1):1–8
27. Egger DJ, Gutiérrez RG, Mestre JC, Woerner S (2020) Credit risk analysis using quantum computers. IEEE Trans Comput 70(12):2136–2145
28. Egger DJ, 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 Trans Quantum Eng 1:1–24
29. Endo S, Sun J, Li Y, Benjamin SC, Yuan X (2020) Variational quantum simulation of general processes. Phys Rev Lett 125(1):010501
30. Fontanela F, Jacquier A, Oumgari M (2021) A quantum algorithm for linear PDEs arising in finance. arXiv:1912.02753
31. García-Ripoll JJ (2021) Quantum-inspired algorithms for multivariate analysis: from interpolation to partial differential equations. Quantum 5:431
32. Giles MB (2015) Multilevel Monte Carlo methods. Acta Numerica 24:259–328
33. Gilyén A, Lloyd S, Tang E (2018) Quantum-inspired low-rank stochastic regression with logarithmic dependence on the dimension
34. Giurgica-Tiron T, Kerenidis I, Labib F, Prakash A, Zeng W (2020) Low depth algorithms for quantum amplitude estimation. arXiv:2012.03348
35. Glasserman P (2004) Monte Carlo methods in financial engineering. Springer, New York
36. Gonzalez-Conde J, Rodríguez-Rozas A, Solano E, Sanz M (2021) Pricing financial derivatives with exponential quantum speedup. arXiv:2101.04023
37. Goodfellow IJ, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y (2014) Generative adversarial networks. arXiv:1406.2661
38. Goubault de Brugière T (2020) Methods for optimizing the synthesis of quantum circuits. Université Paris-Saclay, Theses
39. Grinko D, Gacon J, Zoufal C, Woerner S (2021) Iterative quantum amplitude estimation. npj Quantum Information. 7(1):1–6.
40. Grover L, Rudolph T (2002) Creating superpositions that correspond to efficiently integrable probability distributions. arXiv:0208112
41. Grover LK (1996) A fast quantum mechanical algorithm for database search. In: Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing, STOC '96, New York, NY, USA. Association for Computing Machinery, pp 212–219
42. Hao W, Lefèvre C, Tamturk M, Utev S (2019) Quantum option pricing and data analysis. Quant Finan Econ 3(3):490–507
43. Harrow AW, Hassidim A, Lloyd S (2009) Quantum algorithm for linear systems of equations. Phys Rev Lett 103:150502
44. He C, Li J, Liu W, Peng J, Wang ZJ (2021) A low complexity quantum principal component analysis algorithm. arXiv:2010.00831
45. Herbert S (2021) The problem with Grover-Rudolph state preparation for quantum Monte Carlo. arXiv:2101.02240
46. Heston S (1993) A closed-form solution for options with stochastic volatility with applications to bond and currency options. Rev Finan Stud 6:327–343
47. Hirofumi Nishi TK, Ichiro Matsushita Y (2020) Implementation of quantum imaginary-time evolution method on nisq devices: nonlocal approximation. arXiv:2005.12715

48. A. Holmes and A. Y. Matsuura. Efficient quantum circuits for accurate state preparation of smooth, differentiable functions, 2020. arXiv:2005.04351
49. Huang H-Y, Broughton M, Mohseni M, Babbush R, Boixo S, Neven H, McClean JR (2021) Power of data in quantum machine learning. arXiv:2011.01938
50. Hull J (1997) Options, futures, and other derivatives. Prentice Hall, Hoboken
51. Job J, Adachi S (2020) Systematic comparison of deep belief network training using quantum annealing vs. classical techniques. arXiv:2009.00134
52. Kaneko K, Miyamoto K, Takeda N, Yoshino K (2020) Quantum pricing with a smile: Implementation of local volatility model on quantum computer. arXiv:2007.01467
53. Kaneko K, Miyamoto K, Takeda N, Yoshino K (2020) Quantum speedup of Monte Carlo integration in the direction of dimension and its application to finance. arXiv:2011.02165
54. Kaye P, Mosca M (2004) Quantum networks for generating arbitrary quantum states. arXiv:0407102
55. Kerenidis I, Prakash A (2017) Quantum recommendation systems. In C. H. Papadimitriou, editor, Proceedings of 8th Innovations in Theoretical Computer Science Conference (ITCS 2017), volume 67 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pp 49:1–49:21, Dagstuhl, Germany. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik
56. Kloeden PE, Platen E (2013) Numerical solution of stochastic differential equations. Springer, New York
57. Kubo K, Nakagawa YO, Endo S, Nagayama S (2020) Variational quantum simulations of stochastic differential equations. arXiv:2012.04429
58. Kyriienko O, Paine AE, Elfving VE (2021) Solving nonlinear differential equations with differentiable quantum circuits. Phys Rev A 103(5):052416
59. 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
60. Lloyd S, Mohseni M, Rebentrost P (2014) Quantum principal component analysis. Nature Physics 10:631–633
61. F. A. Longstaff and E. S. Schwartz. Valuing American options by simulation. A simple least-squares approach. *Review of Financial Studies*, 14:113–147, 2001
62. Magniez F, Nayak A, Roland J, Santha M (2011) Search via quantum walk. SIAM Journal on Computing 40(1):142–164
63. S. McArdle, T. Jones, S. Endo, Y. Li, S. C. Benjamin, and X. Yuan. Variational ansatz-based quantum simulation of imaginary time evolution. *Quantum Information*, 5(75), Sept. 2019
64. A. J. McNeil, R. Frey, and P. Embrechts. *Quantitative risk management: concepts, techniques and tools*. Princeton University Press, revised edition, 2015
65. Merton RC (1974) On the pricing of corporate debt: The risk structure of interest rates. The Journal of Finance 29(2):449–470
66. T. Mikosch. *Elementary stochastic calculus : with finance in view*. Advanced series on statistical science & applied probability 6. World Scientifc, London, 1998
67. Montanaro A (2015) Quantum speedup of Monte Carlo methods. Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences 471(2181):20150301
68. K. Nakaji, S. Uno, Y. Suzuki, R. Raymond, T. Onodera, T. Tanaka, H. Tezuka, N. Mitsuda, and N. Yamamoto. Approximate amplitude encoding in shallow parameterized quantum circuits and its application to financial market indicator, 2021. arXiv:2103.13211
69. Orrell D (2020) A quantum walk model of financial options. Capital Markets - Asset Pricing eJournal, Econometric Modeling
70. Orús R, Mugel S, Lizaso E (2019) Quantum computing for finance: Overview and prospects. Reviews in Physics 4:1–13
71. S. Ramos-Calderer, A. Pérez-Salinas, D. García-Martín, C. Bravo-Prieto, J. Cortada, J. Planagumà, and J. I. Latorre. Quantum unary approach to option pricing, 2020. arXiv:1912.01618
72. Ran S-J (2020) Encoding of matrix product states into quantum circuits of one- and two-qubit gates. Physical Review A 101:032310
73. P. Rebentrost, B. Gupt, and T. R. Bromley. Quantum computational finance: Monte Carlo pricing of financial derivatives. *Physical Review A*, 98(2), Aug 2018
74. Reddy P, Bhattacherjee AB (2021) A hybrid quantum regression model for the prediction of molecular atomization energies. Machine Learning: Science and Technology 2(2):025019
75. Romero J, Aspuru-Guzik A (2021) Variational quantum generators: Generative adversarial quantum machine learning for continuous distributions. Advanced Quantum Technologies 4(1):2000003
76. C. Schön, E. Solano, F. Verstraete, J. I. Cirac, and M. M. Wolf. Sequential generation of entangled multiqubit states. *Physical Review Letters*, 95(11), Sep 2005
77. M. Schuld. Supervised quantum machine learning models are kernel methods, 2021. arXiv:2101.11020
78. Schuld M, Sinayskiy I, Petruccione F (2016) Prediction by linear regression on a quantum computer. Physical Review A 94:022342
79. Shao C (2020) Data classification by quantum radial-basis-function networks. Physical Review A 102:042418
80. 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
81. N. H. Shimada and T. Hachisuka. Quantum coin method for numerical integration, 2020. arXiv:1910.00263
82. Soklakov AN, Schack R (2006) Efficient state preparation for a register of quantum bits. Physical Review A 73:012307
83. Stamatopoulos N, Egger DJ, Sun Y, Zoufal C, Iten R, Shen N, Woerner S (2020) Option pricing using quantum computers. Quantum 4:291
84. Y. Suzuki, S. Uno, R. Raymond, T. Tanaka, T. Onodera, and N. Yamamoto. Amplitude estimation without phase estimation. *Quantum Information Processing*, 19(2), Jan 2020
85. K. C. Tan. Fast quantum imaginary time evolution, 2020. arXiv:2009.12239
86. E. Tang. Quantum principal component analysis only achieves an exponential speedup because of its state preparation assumptions. *Physical Review Letters*, 127(6), Aug 2021
87. C. Vázquez. An introduction to Black-Scholes modeling and numerical methods in derivatives pricing. *MAT-Serie A, Universidad Austral*, 17, 2010
88. Vinci W, Buffoni L, Sadeghi H, Khoshaman A, Andriyash E, Amin MH (2020) A path towards quantum advantage in training deep generative models with quantum annealers. Machine Learning: Science and Technology 1(4):045028
89. Wang G (2017) Quantum algorithm for linear regression. Physical Review A 96:012335
90. Wiebe N, Braun D, Lloyd S (2012) Quantum algorithm for data fitting. Physical Review Letters 109:050505
91. Wilmott P (2007) Paul Wilmott Introduces Quantitative Finance, 2nd edn. Wiley-Interscience, USA
92. 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
93. M. Zaheer, C.-l. Li, B. Póczos, and R. Salakhutdinov. GAN connoisseur: Can GANs learn simple 1D parametric distributions? In *Proceedings of the 31st Conference on Neural Information Processing Systems*, NIPS 2017, pages 1–6, 57 Morehouse Lane, Red Hook, NY, US, 2017. Curran Associates Inc
94. Zhao Z, Fitzsimons JK, Fitzsimons JF (2019) Quantum-assisted Gaussian process regression. Physical Review A 99:052331

95. C. Zoufal, A. Lucchi, and S. Woerner. Quantum generative adversarial networks for learning and loading random distributions. *Quantum Information*, 7(103), 2019