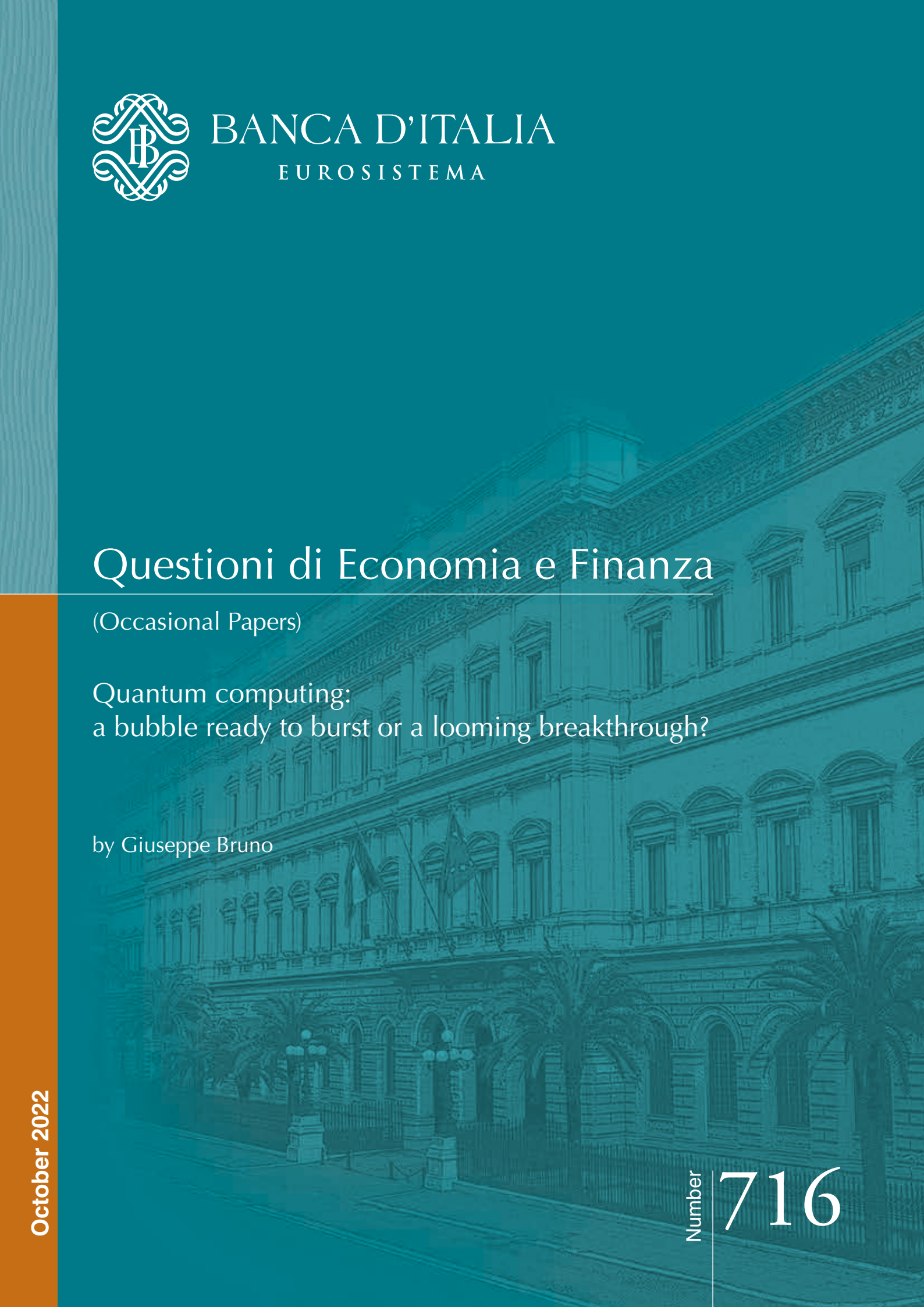
BANCA D'ITALIA
EUROSISTEMA



Questioni di Economia e Finanza

(Occasional Papers)

Quantum computing: a bubble ready to burst or a looming breakthrough?

by Giuseppe Bruno

October 2022

Number 716

BANCA D'ITALIA
EUROSISTEMA

Questioni di Economia e Finanza

(Occasional Papers)

Quantum computing: a bubble ready to burst or a looming breakthrough?

by Giuseppe Bruno

Number 716 – October 2022

The series Occasional Papers *presents studies and documents on issues pertaining to the institutional tasks of the Bank of Italy and the Eurosystem. The* Occasional Papers *appear alongside the* Working Papers *series which are specifically aimed at providing original contributions to economic research.*

The Occasional Papers *include studies conducted within the Bank of Italy, sometimes in cooperation with the Eurosystem or other institutions. The views expressed in the studies are those of the authors and do not involve the responsibility of the institutions to which they belong.*

The series is available online at www.bancaditalia.it .

ISSN 1972-6627 (print)
ISSN 1972-6643 (online)

Printed by the Printing and Publishing Division of the Bank of Italy

QUANTUM COMPUTING: A BUBBLE READY TO BURST OR A LOOMING BREAKTHROUGH?

by Giuseppe Bruno [*]

Abstract

The advent of quantum computation and quantum information theory and the ever increasing empirical possibilities of translating these theories into real physical systems has raised expectations in the private and public sectors. Quantum computers process information using the laws of quantum mechanics. By exploiting superposition (an object can be in different states at the same time) and entanglement (different objects can be deeply connected without any direct physical interaction) quantum computers are heralded as the next technological breakthrough. Compared to traditional digital computing, quantum computing offers the potential to dramatically reduce both execution time and energy consumption. However, quantum algorithms cannot be fully realized on an actual scale of less than 1,000 qubits. The greatest hurdle in harnessing quantum computing is the instability of their quantum mechanical features. Meanwhile, research has shifted towards making "noisy" quantum computers useful. In this work we show three noteworthy applications for central banking banking activities such as gauging financial risk, credit scoring and transaction settlement. These are still proof-of-concepts applications but demonstrate the new software paradigms along with looming potential breakthroughs. We provide a few hints in the trade-off between deploying the innovative technology before it is mainstream and the risk of holding off on adopting it and being surpassed by nimbler competition.

JEL Classification: C65, C87.
Keywords: quantum computing, quantum information, superposition, entanglement.
DOI: 10.32057/0.QEF.2022.0716

Contents

[*] Bank of Italy, Directorate General Economics, Statistics and Research.

1 Introduction

Nature isn't classical dammit, and if you want to make a simulation of Nature you better make it quantum mechanical, and by golly it's a wonderful problem because it doesnt look so easy.

– Richard Phillip Feyman, 1981

In May 1981, in a conference on the topic *Simulating physics with computers*, Feynman, 1965 Nobel laureate in physics, explained and envisaged that digital computers would be unsuitable for simulating the behavior of quantum system, for example see Preskill 2021 [26].

In these four decades computing resources have kept growing in power following what is called the *Moore's law* [23], which states that the computer power doubles for constant cost roughly every two years. While hardware designers grapple with the demise of Moore's law, prototypes of a completely new type of machine - the quantum computer - have been introduced. These devices exploit the properties of quantum mechanics in particular, phenomena known as superposition and entanglement to speed up certain classes of calculations.

Even though the scale of actual quantum computers is relatively limited, we can now witness a new generation of quantum algorithms which only require very limited resources and robustness against errors. This gives rise to the so-called Noisy Intermediate Scale Quantum computers (NISQ) era. A promising group of algorithms and methods that overcome at least some limitations in the NISQ-era are the so-called hybrid quantum-classical algorithms, or variational quantum algorithms. In general, these quantum algorithms have free parameters and other tunable parts that run on quantum hardware, but they are (partially) controlled using classical computation, thus the term hybrid is used. Comparable to other specialized hardware such as Graphical Processing Units (GPU), in this setting, a Quantum Processing Unit (QPU) is considered as a computational resource which can be leveraged for certain parts of an algorithm that benefit from the potential speedup or resource efficiency. Here we take into account three of the most relevant applications for Central Banking activities and the banking industry as a whole:

1. Financial risk evaluation such as *Value at Risk* and *Conditional Value at Risk*;
2. credit scoring through Variational Quantum circuits;
3. transaction settlement problem for securities settlement in capital markets.

In the empirical sections we explore quantum algorithms designed to solve these kind of problems. Quantum computers promise to provide a quadratic speed-up over classical Monte-Carlo simulations which may be used to evaluate risk, see Woerner et al. 2019 [34] and Egger et al. [10] and price financial derivatives (for example see Rebentrost 2018 [27]). The banking industry has already begun its analysis of quantum platforms, see for example Egger et al. 2020 [11]. Although we are in an infancy stage, it is of paramount importance to closely follow the coming steps in the quantum technology evolution.

The rest of the paper is organized in the following way. After this introduction, section 2 surveys the most relevant hardware solutions available on the market. Section 3 sketches the main open source solutions for running quantum algorithms. The following three sections describe the chosen examples: section 4 presents an example of the evaluation of the credit risk for a portfolio of loans, section 5 shows a quantum machine learning application for gauging credit scoring on a publicly available dataset and section 6 provides an example of a combinatorial optimization applied to a capital market infrastructure. Finally section 7 provides some concluding remarks and suggests future avenues of further research.

2 The commercially available hardware solutions

In the last fifteen years many steps forward have been made in the development of quantum computing hardware based on different physical systems. Swift progress has been fuelled by the assumption that sufficiently powerful quantum machines will yield huge computational advantages in many fields such as new drugs discovery, machine learning, chemistry, combinatorial optimization, risk analysis and quantum security. The most relevant physical and logical element of a quantum computer is the qubit [1]. As quantum devices, qubits show the following features:

1. superposition, which allows a single qubit to be in different states at the same time;
2. entanglement, which consists in the possibility to prepare two or more qubits in a strictly correlated fashion where modification of the status of one qubit instantaneously causes the modification of the status of another qubit;
3. decoherence over time, i.e. the disappearance of the superposition of quantum states [2].

The general requirements for qubit design have been well summarized in Di Vincenzo (2000) [9] by the following 5 items:

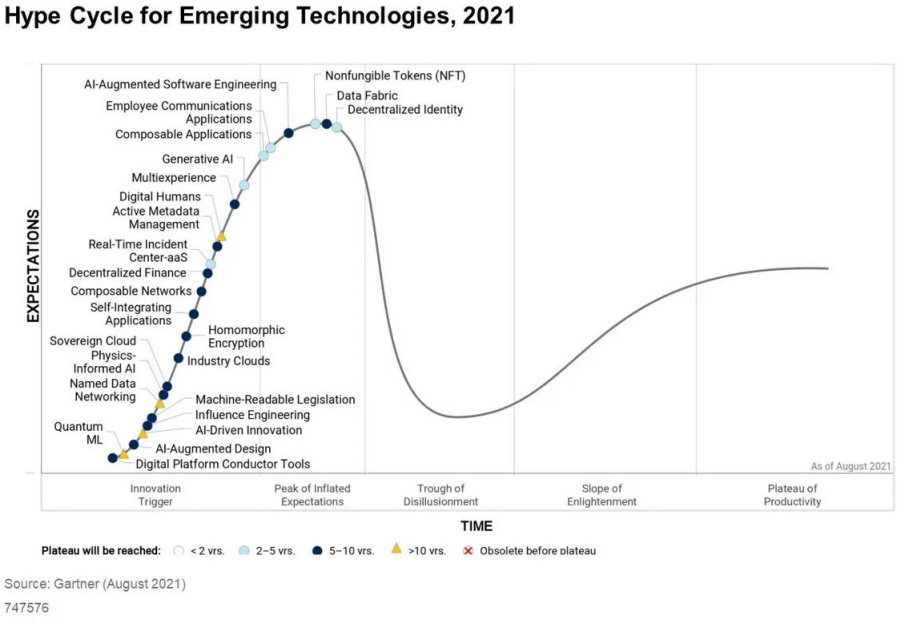
1. a scalable system with well-characterized qubits;
2. ability to initialize each qubit (for computation);
3. stability of qubits (i.e., long decoherence times)
4. support for universal instruction set for arbitrary computation
5. ability to measure qubits (i.e., readout in computational basis)

These 5 requisites are unfortunately in conflict with one another. Specifically, initializing and performing computations on a qubit require interactions on the system, which will inherently break the required conditions for the realization of a stable qubit. This is one of the reasons why it is fundamentally difficult to build a quantum computer.

[1] This represents simultaneously a quantum system with two orthogonal states and the unit of information in quantum computing.

[2] Nowadays decoherence times are within the range $10^{-9} \div 10^{-3}$ seconds.

Notwithstanding the growing interest in quantum computing, there have yet to be any fundamentally new results due to quantum computing [3]. Most of the new technologies follow the so called hype cycle pattern, see Gartner (2020) [14]. In the following picture we show the Gartner's 2021 hype cycle for the more relevant emerging technologies.



Source: Gartner (August 2021)

747576

Here, the Quantum machine learning applications appear to be at the very beginning of the quickly growing **innovation trigger** phase and they are marked as those requiring more than 10 years before extensive adoption.

The most relevant hurdles that currently stifle the development of quantum computers are limits to the number of computational units and computational errors.

In order to take full advantage of quantum computing in a meaningful way, a technological leap is needed to reach at least hundreds of thousands of physical qubits. This is required if one needs, for example, to optimize large portfolios or to accurately estimates the Value at risk for a multi-loan package [4] . In stark contrast with this requirement, today's quantum computers rely on a small number of noisy qubits (order of 10^2) because the qubits manufacturing technology is not yet sufficiently developed for larger scales (see for example Wang 2021 [33]). For this reason, present quantum processors are dubbed **NISQ** (Noisy Intermediate-Scale Quantum) (see Preskill (2018) [25]). The necessity to define a noisy processor stems from the intrinsically analog nature of a qubit. This is at complete variance with digital computers which employs two well separated level of a physical variable

[3] In October 2019 Google announced quantum supremacy in an Nature article [2]

[4] Check for example Ray LaPierre 2021 [22] *Introduction to Quantum Computing*

to represent one of two binary values [5]. At least in the present decade, quantum computers are likely to remain specialized devices, which will be accessed via the cloud. Nonetheless, quantum technology might be preferred even if classical supercomputers ran faster, if, for example, the quantum hardware had lower cost and power consumption. In any case, we should take into account that the power of classical computers keep increasing. In 2021, Fugaku, the fastest computing platform has achieved around $0.5 \cdot 10^{18}$ FLOPS for linear algebra tasks[6]. Quantum computers fight to catch up with a moving target. Classical hardware and algorithms keep improving steadily. A quite popular figure of merit for NISQ systems is called quantum volume (**QV**). It summarizes in one single metric the number of available qubits, their fidelity [7] and how extensively they are interconnected [6].

The following table 1 shows a summary of the main quantum computers available on the market at the end of 2021.

Table 1 Quantum Computing Hardware already available.

Manufacturer	**Technology**	**Max # Qubits**	Gate Fidelity	Quantum Volume	**Cloud Access**
AQT	Trapped Ions	20 (Eagle)	99.9	N.A.	yes
Coldquanta	Cold Atom	100 (Hilbert)	99.1 and 95.0	N.A.	no
Google	Superconducting	53	99.3	N.A.	yes
IBM	Superconducting	127 (Eagle)	99.1	64	yes
Rigetti	Superconducting	40 (Aspen 11)	99.6 and 90	N.A.	yes
Honeywell	Trapped Ions	10 (H1)	99.9 and 99.1	1024	N.A.
IonQ	Trapped Ions	11	99.9	$4 \cdot 10^6$	yes
QuTech	Silicon	2 Spin2 QPU	99.5	N.A.	yes
Xanadu	Photonic	24	N.A.	N.A.	yes

The high number of players and differences in qubit implementation are a witness of a market that has yet to identify the best technological solution.

While classical computers are invariably made with silicon/germanium semiconductor technology, Quantum Processing Units (QPU) can also be realised with superconductor chips, ions or neutral atoms trapped in a vacuum, on-chip photonic waveguides and other solid state devices. These solutions provide different tradeoffs in terms of number and fidelity of qubits, phase coherence time, connectivity etc. This landscape is not new for the relationship between technology and market structure. As it was clearly expressed in Varian (2001) [31], highly innovative industries are subject to the same market forces as every other industry. This means that once the quantum technology will reach maturity the cheapest or best fit will survive leaving only few technological solutions which will be adopted in the consumer market. A different situation is emerging in the available software solutions for programming quantum computers. This is described in the following section.

[5] For example the current or the voltage in selected points of the circuit

[6] Floating Points Operations per second

[7] Qubit fidelity is the degree of confidence in the results of a quantum computation

3 Quantum Algorithms and the developed software solutions

With the rise of the first quantum computers, suitable programming languages and quantum algorithms came up with promising results.

Quantum algorithms started to come out in 1994 with the work of Shor [29] and [30] who proposed a much more efficient integer factorization based on the period finding algorithm. In 1996 Grover [16] introduced a quantum algorithm to speed up the unstructured search problem quadratically by employing the quantum amplitude estimation, Harrow et al. 2009 [18] put forward a method for solving system of linear equations characterised by an Hermitian matrix [8]. Nevertheless, quantum software has not begun to be produced in a large-scale, industrial way yet. The initial Quantum software development kits started to come out from in 2017 and the following years. Later we have seen other software frameworks such as Pennylane and Ocean.

These software packages and others have been instrumental for testing other noteworthy algorithms such as VQE (Variational Quantum EigenSolvers) and QAOA (Quantum Approximate Optimization Algorithms). These algorithms are heuristics designed for near-term, noisy quantum computers without performance guarantees (see Fahri 2014) [12].

All the empirical applications suitable of benefitting from quantum computing cannot be accomplished uniquely with quantum computing hardware. They need quantum software and suitable algorithms which are able to take advantage of the laws of quantum mechanichs. One of the main takeaway of this work is that everyone involved in the software development business will have to take into account new software development life cycles. Quantum algorithms are usually described in a high-level language (e.g. Python or C++). They are then *translated* into a quantum circuit consisting of a series of quantum gates applied in a sequential manner [9]. These circuits get translated into Quantum Assembly (QASM) [10] and executed in real devices or simulators. The final step is given by a measurement which collapses the quantum state into a classical one from which the result of the algorithm is being inferred via classical post-processing techniques. At the present time, the area of quantum software development is relatively new and less established than that of quantum hardware. Starting from 2017, quantum software tools are being developed at a rapid pace with many packages now available from different platforms such as D-Wave, Google, IBM, Microsoft and Xanadu. These software tools are able to operate at relatively low level such as at the assembly language. They all realize a three phase design flow which maps a high-level program representation

[8] A matrix X is defined hermitian when it is equal to its conjugate transpose $X^{t*} = X$

[9] The quantum gate model is just one of the computing model. There are others, such as Measurement-Based Quantum Computing and Adiabatic Quantum Computing (D-Wave). However, currently the gate model seems to be that one more likely to be employed in full-scale fault-tolerant quantum computation

[10] Quantum Assembly Language (QASM) is an extension of a RISC assembly language with the classical RISC instruction set integrated by a set of platform dependent quantum instructions.

of a quantum algorithm into a hardware bound realization or a software classical simulation (see figure 1).

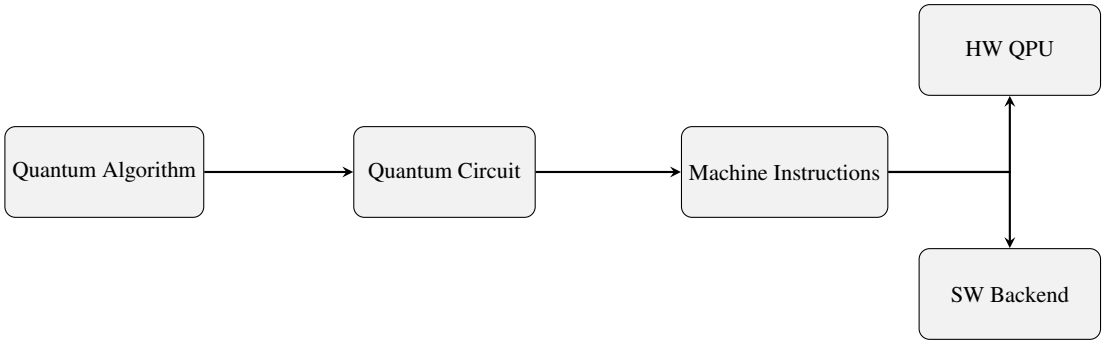


Fig. 1 Conceptual Quantum software design flow.

See Chong et al. (2017) [5] for a deeper analysis. Today different quantum computing models are available. Among them we mention the most relevant ones:

- gate-based (cfr. Kwon (2021) [21]);
- measurement-based (see Jozsa (2005) [20]);
- and adiabatic quantum computing (see Aharonov at al. (2008) [1]).

In this work we have written and tested code running the Qiskit and Pennylane quantum software development kits [11].

Qiskit is an open-source compilation framework targeting various types of hardware and a high-performance quantum computer simulator with emulation capabilities. It was designed by IBM Research to allow software development for their cloud quantum computing service.

Pennylane is a cross-platform Python library for quantum machine learning, automatic differentiation, and optimization of hybrid quantum-classical computations. The software development of quantum algorithms follows the scheme shown in the picture 2:

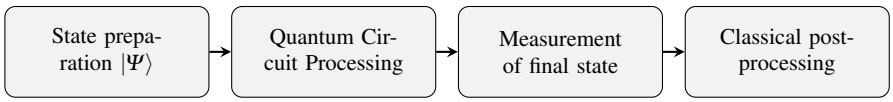


Fig. 2 Quantum software development cycle.

The following sections describe the three considered empirical applications, while the different coding examples are left in the appendix.

[11] The examples have been run on the different Quantum machines kindly provided by the IBM Canada.

4 Financial Risk Evaluation

Risk capital models are crucial tools for the financial sector. Among them we mention capital allocation, performance evaluation, risk pricing, risk identification and monitoring, and capital adequacy assessment. Under the Basel II Accord, Financial Intermediaries (FI) are required to assess capital adequacy for credit, market and operational risks. FIs determine regulatory capital for credit risk usually using internal rating based (IRB) approach. See [24] for reference. In this section, we restrict our analysis to the basic Gaussian Conditional Independence Model proposed by Vasicek in [32] which transforms unconditional Probability of Default (PD)s into PDs conditional on a single systematic risk factor. In particular assuming a portfolio of K assets sharing a common expiration time T, denoting by λ_i the loss given default on the ith loan and by $X_i(Z)$ a binomially distributed variable depending on the common random variable Z, we have

$$L = \sum_{i=1}^{K} \lambda_i \cdot X_i(Z) \tag{1}$$

for the portfolio loss. Assuming independence among the loan defaults, by the central limit theorem the portfolio would asymptotically converge to a normal distribution. The lack of independence prevents the unconditional normal distribution but adopting the Vasicek assumption, conditionally on the common factor (Z), the default event are independent and equally distributed. By the law of large numbers, in this circumstance, the portfolio loss converges to its expectations and it is possible to derive the following closed form solution for the cumulative distribution function of loan losses on asymptotically large portfolios

$$P[L < x] = \Phi\left(\frac{\Phi^{-1}(x) - \sqrt{\rho}\Phi^{-1}(\alpha)}{\sqrt{1-\rho}}\right) \tag{2}$$

where Φ represents the Gaussian standard cumulative distribution function, α is a confidence level and $\rho \in [0,1)$ is the assumed constant pair-wise correlation among the assets. For Risk management purposes we are interested in evaluating the Value at Risk **VaR** and Conditional Value at Risk **CVaR**. These two figures are are quantiles of the loss distribution which have an utmost influence on the computation of regulatory and economic capital for assessing market risk.

Their definitions are reported in the following:

$$\mathbf{VaR}_\alpha(L) = \inf\{x \in \mathbb{R} : \mathbb{P}[L \leq x] \geq 1-\alpha\} \tag{3}$$

$$\mathbf{CVaR}_\alpha(L) = \mathbb{E}[L : L \geq \mathbf{VaR}_\alpha(L)] \tag{4}$$

Monte Carlo simulation is the classical method employed to determine **VaR** and **CVaR**, see for example Jorion (2001) [19]. Monte Carlo is based on the computation of the time evolution of the portfolio assets for N different realizations and computing its aggregated value. Following this avenue **VaR** calculations features

confidence interval which scales as $\mathbb{O}(N^{-1/2})$. By harnessing the principle of quantum superposition it is possible to estimate the **VaR** or **CVaR** of the given portfolio with a theoretical quantum improvement which scales as $\mathbb{O}(N^{-1})$ which represents a quadratic speed up over the Monte Carlo procedure which goes as $\mathbb{O}(N^{-1/2})$. An open source toy example has been taken from the Qiskit repository and a simple benchmark has been realized. The key element in the **VaR** and **CVaR** estimation is the employment of the algorithm dubbed Quantum Amplitude Estimation (QAE) which is a generalization of the widely known Grover's searching algorithm (see for example see Grover 1996 [16] and 1997 [17]). Here, by suitably superposing all the states where we have a default we are able to compute the quantiles of the loss distribution with a quadratic performance improvement over the Monte Carlo procedure. Assuming we have a portfolio of $n = \log_2(N)$ assets we could prepare our initial quantum state:

$$|\Psi\rangle_n = \sum_{i=0}^{N-1} \sqrt{p_i} \cdot |i\rangle_n \tag{5}$$

where: p_i is the probability of measuring state $|i\rangle_n$ [12] which is the binary encoding of the defaulted and non defaulted assets in our portfolio [13]. We then introduce the function $\phi : \{0,\ldots,N-1\} \to [0,1]$ and the quantum operator

$$F : |i\rangle_n |0\rangle \to |i\rangle_n (\sqrt{1-\phi(i)}\,|0\rangle + \sqrt{\phi(i)}\,|1\rangle) \tag{6}$$

At this point we apply this operator to the quantum state in equation 5 to go to state:

$$\sum_{i=0}^{N-1} \sqrt{1-\phi(i)}\sqrt{p_i}\,|i\rangle_n |0\rangle + \sum_{i=0}^{N-1} \sqrt{\phi(i)}\sqrt{p_i}\,|i\rangle_n |1\rangle \tag{7}$$

By choosing $\phi(i) = \frac{X}{N}$ allows us to estimate the expectation of our unknown distribution by evaluating the probability of measuring a $|1\rangle$ in the last qubit. To find the $VaR_\alpha(X)$ we employ the function

$$f_k(i) = \begin{cases} 1 \text{ if } i < k \\ 0 \text{ if } i \geq k \end{cases} \tag{8}$$

Applying the quantum operator 6 to the quantum state in equation 5 to get:

$$\sum_{i=k+1}^{N-1} \sqrt{1-\phi(i)}\sqrt{p_i}\,|i\rangle_n |0\rangle + \sum_{i=0}^{k} \sqrt{\phi(i)}\sqrt{p_i}\,|i\rangle_n |1\rangle \tag{9}$$

The probability of measuring $|1\rangle$ in the last qubit is $\sum_{i=0}^{k} p_i = \mathbf{P}[X \leq k]$. By running a search over k we find the $\mathbf{Var}_\alpha(X)$ as the smallest value k_α for which $(P)[X \leq k_\alpha] \geq 1-\alpha$. In a similar way, by picking a suitable function $f(i)$ we com-

[12] for an explanation of the Dirac $|\cdot\rangle$ notation see the Appendix A

[13] With $N = 4$ we have $|0\rangle_2 = |0,0\rangle_2$ which represents non defaults, while $|2\rangle_2 = |1,0\rangle_2$ represents a defaults in the second asset.

pute also the $\mathbf{CVar}_\alpha$. In Appendix B we provide some code snippets and present some results.

5 Credit Scoring with Variational Quantum circuits.

Despite the recent explosion in the availability of a wide array of banking services, lending still constitutes a significant source of income for commercial banks. The lending process is, in general, a relatively straightforward series of steps involving the borrower and the lender. These activities range from the initial loan application to the successful or unsuccessful repayment of the loan. Although retail lending belongs among the most profitable investments in lenders' asset portfolios, increases in the number and amounts of loans bring increases in the number of defaulted loans. Thus, the primary problem of any lender is to accurately distinguish between *good* vs *bad* borrowers prior to granting credit. Therefore credit scoring is commonly recognized to help in boosting a country economic growth. Moreover it is a valuable tool for improving financial inclusion, individual credit access and efficiency. The wide adoption of innovative methods for credit scoring raises concerns about data privacy, fairness and explainability of the models. Though these concerns should not stifle innovation, especially if it hinders improvements in financial inclusion and risk assessments. In this section, we show how we can combine a classical and quantum computing platform to speed up the process of credit scoring. In the previous sections we have seen how quantum computing can substantially speed-up particular tasks. Search in an unstructured database runs quadratically faster than the classical limit. Quantum prime factorization could be performed much more efficiently, enabling to break the current public cryptography securing our e-commerce/banking activities [14]. In this section we consider whether and how quantum computing can improve or accelerate machine learning tasks. In the affirmative case, what kinds of speed-ups can we expect over the best classical algorithms? Could it also lead to a better generalization/learning performance? For answering to these questions we have analysed the credit scoring problem with the *German credit dataset* [15].

A quite interesting group of quantum algorithms and methods which are able to cope with at least some limitations in our the NISQ-era are the so-called hybrid quantum-classical algorithms. In general, these quantum algorithms have free parameters which are optimized on quantum hardware, but they are in part controlled using classical computation, thus the term hybrid is used. This variational algorithms can be schematically described by the following picture: For classifying the borrowers we are confronted with a standard supervised machine learning problem. We have an array of D feature data $x_i \in R^D$ with $i = 1, \ldots, N$ and the corre-

[14] The present record, published in 2018 [7], is the composite number 4,088,459 which has 7 digits but the quantum algorithm doesn't generalize to all the 7 digits numbers. Current RSA key are composite number of 617 decimal digits.

[15] The data set consists of loan applications from 1,000 individuals and is described at https://www.kaggle.com/datasets/uciml/german-credit

sponding true labels $y_i \in \{0,1\}$. The algorithm will estimate a classification function $\hat{y}_i = f(x_i, \theta)$ by choosing the best vector parameter $\hat{\theta}$ which minimise the total error $\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$.

The classification function is estimated by a quantum kernel which is just an extension of the classical kernel method. Quite often feature data cannot be easily separated by a hyperplane in its original space. A commonly used technique consists in looking for such a hyperplane in a higher dimensional feature space by applying a non-linear transformation function to the data.

Classifying our observations in this new feature space is nothing more than evaluating how close data points are to each other. This is carried out by computing the inner product for each pair of data points. So we do not need to compute the non-linear feature map for each datum, but only the inner product of each pair of data points in the new feature space (see for example Shoelkopf (1998) [3]). Therefore instead of computing the actual feature maps we can simply compute a kernel function derived from the inner product:

$$K(x,x') = |\langle \Phi(x), \Phi(x') \rangle| \tag{10}$$

where $x, x' \in R^D$ are two different features vectors whereas $\Phi(\cdot)$ is the corresponding feature map. It is frequent the case where feature maps are hard to compute while their kernels, which, in our case, means their inner product, are straightforward. The feature map $\Phi(\cdot)$ maps the original measurable properties of the phenomenon under study into a new space which will enable an easier classification.

Here we will address this Machine Learning problem by means of a quantum linear model instead of the classic SVM (Support Vector Machines), logistic regression or even a gradient boosting algorithm. In our quantum example, the kernel method refers to the encoding strategy of the original data into a state superposition. For every original data point $x_i \in R^D$ we will have the following encoding:

$$|\psi_x\rangle = \sum_{n=1}^{D} |i\rangle \tag{11}$$

This solution is quite convenient because, in the quantum framework, a size D vector requires just $log_2(D)$ qubits. The state vector $|\psi_x\rangle$ presents a superposition of all the possible features. This explains why we can achieve an exponential speed up respect to the problem size when we prepare the state into this superposition.

5.1 The Variational Quantum approach

Variational quantum circuits are a family of hybrid quantum-classical algorithms that iteratively carry out operations on a quantum platform and successively on a classical one. The quantum gates in these circuits behave like successive processing

layers. The key feature of these schemes is that all quantum operations are linear and unitary transforms. The sole non-linearity operation is the final measurement which has the effect of collapsing any quantum state into a classical value. The variational quantum architecture for our NISQ computation is based on a short-depth circuit where each feature is suitably encoded into a quantum register. The output of the variational circuit consists in some expectation values over multiple qubit measurements of the resulting quantum state $|\psi_x'\rangle$. This output is fed into the objective function which is usually the sum of the squared errors computed in the training sample. This objective function is optimized on a classical computer with respect to the set of parameters θ which completely defines the behaviour of the quantum variational circuit. Therefore, in this algorithm we iterate between the quantum and the classical computation until we reach convergence between the classification error and the parameter vector θ of the quantum variational circuit. The empirical results show that the quantum machine learning algorithm provides a classification function which achieve 87 % of accuracy by employing the quantum feature maps available in the software framework. All the numerical outcomes are available in Appendix C.

6 Transaction Settlement Optimization

Securities settlement is one of the most relevant functions in a centralized capital market infrastructure. A securities settlement system (**SSS**) executes settlement orders based on incoming settlement instructions (**SI**), arising from the trading activity. A **SI** usually includes two corresponding transfers: securities and cash for the delivery and payment of securities respectively. Delivery must occur if and only if payment occurs. This entails that the **SSS** must either settle both transfers in the instruction or none of them. This corresponds to the principle of delivery versus payment (**DVP**) and the relative **SI** are defined as the **DVP** instruction.

The settlement is finalized only when the involved parties own the assets that they have agreed to exchange. The problem is that the settlement of a securities transaction does not happen concurrently with the trading activity. Thus, even if the parties have the assets at trading time, they may not at the time of settlement.

During the development of the trading activity, the **SSS** might end up in a gridlock, where a group of settlement instructions can settle only if netted (that is tantamount to aggregate those obligations), whereas no single instruction in the group can settle on its own (see Figure 3 for an example).

With the aim of avoiding such situations, **SSSs** have to solve the task of maximizing number or total value of settlement instructions without overdrawing any participant account.

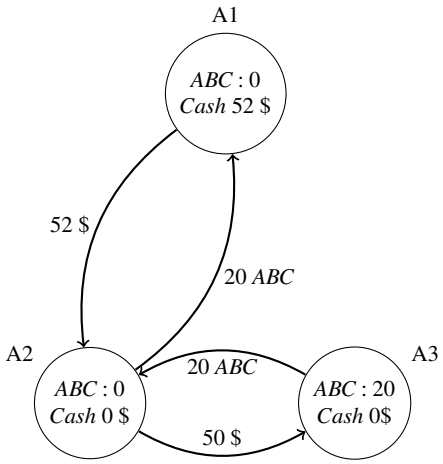


Fig. 3 A settlement gridlock. The circles show the cash and security balances of each party. The arrows indicate the stock ABC transfers of the two settlement instructions. With the initial balances neither of the settlement instructions can be settled, but they can be settled through netting.

6.1 The integer programming problem

Here we present the optimization problem faced by a clearing house receiving continuously trade orders. For any settlement interval the clearing house has all of the received DVP transactions plus all the remaining unsettled transactions remaining from the previous batch. Let n be the total number of transactions in the present batch, the binary variable x_j for $j \in [1,n]$ indicates whether or not the transaction j is settled and w_j represents the weight of settling it. In this case our optimization problem can be set as:

$$\begin{aligned} &\max \sum_{j=1}^{n} w_j \cdot x_j \\ &\text{s.t.} \sum_{j=1}^{n} v_{ij} \leq b_i \quad \text{for} \quad i \in [1,m] \\ &\quad x_j \in \{0,1\} \quad \text{for} \quad j \in [1,n] \end{aligned} \tag{12}$$

where b_i, for each $i \in [1,m]$, is a non-negative integer indicating the size of balance i, $v_{i,j}$ represents the obligation that transaction j puts on the balance i. The weight w_j might be the total monetary value of transaction j, or $w_j = 1 \quad \forall\, i$ if we seek to maximize the whole number of settled transactions. As it is shown, for example, in Shafransky and Doudkin (2006) [28] this problem belong to those whose complexity is *NP-hard*. This categorization is an asymptotic concept which means that an algorithm for solving this problem in polynomial time is not known (see Goldreich (2010) [15] for reference). This circumstance opens

the road for the search of more effective quantum algorithms. Some of these, such as the **VQE** and **QAOA** have been proposed to solve these problems on a quantum computer. Though, so far only toy examples can actually be solved on the presently noisy available platforms. Qiskit example codes are available on github (https://github.com/GiuseppeBruno1959). As empirical example, we have considered the third example presented in Braine et al. (2019) [4].

In this case we consider the following set of 7 transactions among six parties.

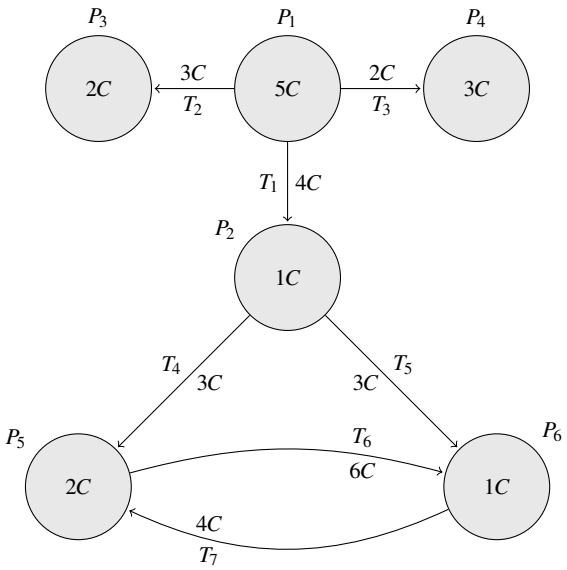


Fig. 4 Transaction scheme showing a batch of seven payments $T_1;\ldots;T_7$ between six parties $P_1;\ldots;P_6$. Each party owns a currency account C, whose balance is shown in the interior circles.

For the instance shown in fig. 4, we coded a sequence of classical and quantum simulation [16] We have tried the ADMM [17] (see for example Gambella 2021 [13]) and the QAOA quantum optimization algorithms [18]. We have reached a global optimal solution settling payments $T_2;T_3;T_6;T_7$ only with the QAOA algoritm [19].

This is simply a toy example which could be compared with around 500 settlement per minute on T2S [20]. The existence of such algorithms, even in their infancy stage, do open new optimization scenarios in the Transaction Settlement problem for delivery versus-payment securities transactions. This is an important optimization challenge in capital markets. In perspective, quantum algorithms could increase settlement efficiency (in terms of the number of transactions settled for a given period), thereby minimizing the time intervals between trade and settlement. In Appendix D we provide some details, code snippets and the computed results for the problem considered.

[16] The formal description of the optimization problem is shown in Appendix D.

[17] The Alternate Direction Method of Multipliers is an augmented Lagrangian method suitable for integer variable optimizations

[18] The two algorithms are available in the IBM Qiskit library.

[19] this corresponds to the bitstring 0110011.

[20] Target 2 Security is the European platform for securities settlements

7 Concluding remarks

In this paper we have presented the current status of Quantum Computing technologies and some of their present practical applications. Expectations are sky-high, but in practice quantum computers are apparently still one decade away from becoming widely available technologies that can deliver significant advantages in the business world. The hardware is trying to catch up with the actual users' needs, and it will take some major breakthroughs for quantum computers to be converted into commercial products that we could use to consistently solve critical, everyday business problems. However this scenario should be seen as an incentive to pursue sound empirical research in this field at least following in the steps of other big financial institutions such as Goldman Sachs (Inside Quantum Technology news April, 30 2021), Bank of Canada (Inside HPC April 14, 2022) and BBVA (research paper with Zapata Computing May 25 2021).

In this work we have shown the application of Quantum algorithms to the following banking problems:

1. *Value at Risk* and *Conditional Value at Risk* through quantum amplitude estimation;
2. Quantum machine learning classification for credit scoring;
3. Optimization for the transaction settlement problem in the capital markets framework.

We have seen how these standard problems can be translated in such a way which becomes suitable for quantum platforms to solve them. Although the development of these Quantum Computing platforms is at the beginning of their path needed in order to become fully fledged industrial production products, it has been shown that a good push to their development will come from the realization of Quantum algorithms and software development tools helping researchers in harnessing the new technologies without the burden of understanding the physical laws and processes behind a quantum hardware device.

1 Appendix A: Notation

In the paper we follow the so called Dirac notation for the linear algebra involved in quantum mechanics [21]. This is a formal language aimed at answering the needs of expressing states in quantum mechanics. The first notion is the *ket* $|\cdot\rangle$ which is a vector with components in $\mathbf{C}$. If we have N basis vectors $|i\rangle\,, \quad i = 1,2,\ldots,N$ then any *ket* $|v\rangle$ can be written as

$$|v\rangle = \sum_{i=1}^{N} v_i\,|i\rangle$$

[21] see Dirac (1939) [8]

The *adjoint* of a *ket* is called a *bra* $\langle v| = |v\rangle^{\dagger}$. The dagger † is the usual notation for the complex conjugate of the transpose. The inner product between a bra and a ket is written as $\langle \cdot | | \cdot \rangle$ and it represents the probability amplitude of a given quantum state.

2 Appendix B: Financial Risk Evaluation

For this example we have considered a portfolio composed of three assets with the following parameters:

```
# set problem parameters
n_z = 2
z_max = 2
z_values = np.linspace(-z_max, z_max, 2 ** n_z)
p_zeros  = [0.15, 0.25, 0.2]
rhos     = [0.1, 0.05, 0.15]
lgd      = [1, 2, 2]
K        = len(p_zeros)
alpha    = 0.05
```

With these parameters we can build the quantum state for the common Gaussian random

$$|\Psi\rangle = \sum_{i=0}^{2^{n_z}-1} \sqrt{p_z^i |z_i\rangle} \bigotimes_{k=1}^{K} \left(\sqrt{1-p_k(z_i)}\,|0\rangle + \sqrt{p_k(z_i)}\,|1\rangle \right) \tag{13}$$

At this point, we can define our Gaussian conditional independent model from a Qiskit library and compute the corresponding values for:

- expected Loss $\mathbf{E}[L]$;
- pdf and cdf of L;
- value at Risk $\mathbf{VaR}(\mathbf{L})$ and its corresponding probabilities;
- Conditional value at Risk $\mathbf{CVaR}(\mathbf{L})$;

These items are computed with the following code:

```
from qiskit_finance.circuit.library import GaussianConditionalIndependenceModel as GCI
u = GCI(n_z, z_max, p_zeros, rhos)
# run the circuit and analyze the results
job = execute(u, backend=Aer.get_backend("statevector_simulator"))
# analyze uncertainty circuit and determine exact solutions
p_z = np.zeros(2 ** n_z)
p_default = np.zeros(K)
*** snippet **
state = job.result().get_statevector()
if not isinstance(state, np.ndarray):
    state = state.data
for i, a in enumerate(state):
    # get binary representation
    b = ("{0:0%sb}" % num_qubits).format(i)
    prob = np.abs(a) ** 2
    # extract value of Z and corresponding probability
    i_normal = int(b[-n_z:], 2)
    p_z[i_normal] += prob
    # determine overall default probability for k
    loss = 0
```

```
    for k in range(K):
        if b[K - k - 1] == "1":
            p_default[k] += prob
            loss += lgd[k]
    values += [loss]
    probabilities += [prob]
values = np.array(values)
probabilities = np.array(probabilities)
expected_loss = np.dot(values, probabilities)
losses = np.sort(np.unique(values))
pdf = np.zeros(len(losses))
for i, v in enumerate(losses):
    pdf[i] += sum(probabilities[values == v])
cdf = np.cumsum(pdf)
i_var = np.argmax(cdf >= 1 - alpha)
exact_var = losses[i_var]
exact_cvar = np.dot(pdf[(i_var + 1) :], losses[(i_var + 1) :]) / sum(pdf[(i_var + 1) :])
```

The results of our example are shown in table 2 :

Expected Loss E[L]:	0.6381
Value at Risk VaR[L]:	2.0000
P[L <= VaR[L]]:	0.9605
Conditional Value at Risk CVaR[L]:	3.0000

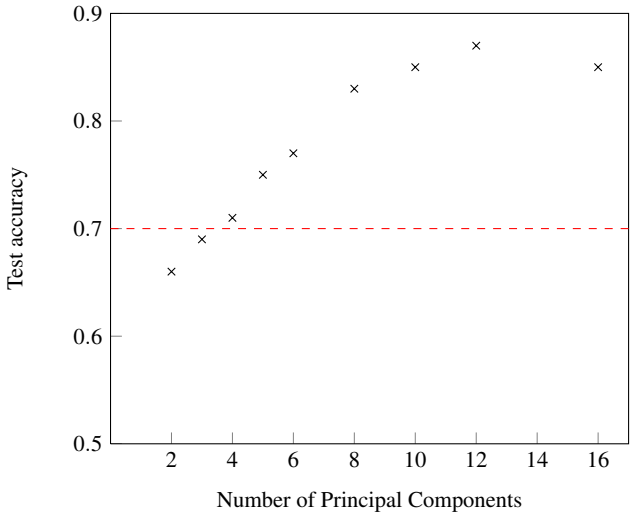
Table 2 Numerical results achieved with the quantum computer

3 Appendix C: Credit Scoring

In this empirical application we have explored the different results achieved by varying the number of input variables by Principal Component Analysis (PCA). This technique comes from linear algebra. Here it is used as a data preparation technique to projection the original dataset into a smaller dimensionality space prior to fitting a model. In the following code snippet we show the main computing loop for running the Quantum support vector machine with a different dimension for the PCA reduction.

```
for i in pca:
    lista_tmp=[]
    for c in list_cost:
        lista_cv=[]
        for y in [2,3]:
          lista_cv.append(get_acc(i,1,c,classic=False,verbose=True,
          simulator='qasm_simulator',maps='Z',
          random_state=y)[0])
        lista_tmp.append(np.mean(lista_cv))
    cost_value=list_cost[np.argmax(lista_tmp)]
    acc_quant_z_1_qasm.append(get_acc(i,1,cost_value,classic=False,
    verbose=True,simulator='qasm_simulator',maps='Z',
    random_state=y,is_final=True)[0])
```

In the figure 3 we show the classification accuracy achieved with the Quantum SVM ran on quantum processor.



As it is shown in the picture the classification accuracy with the quantum support vector machine and the given feature map keeps increasing reaching a maximum of 0.87 with 12 principal components.

4 Appendix D: Transaction Settlement

In this example we have optimized the number of transactions to be settled in a given settlement interval. The integer programming problem taken from [4] is the following:

$$\begin{aligned}
&\max(x_1+x_2+x_3+x_4+x_5+x_6+x_7)\\
&s.t.\\
&C1: 5-4x_1-3x_2-2x_3 \geq 0\\
&C2: 1-4x_1-3x_4-3x_5 \geq 0\\
&C3: 2+3x_2 \geq 0\\
&C4: 3+2x_3 \geq 0\\
&C5: 2+3x_4-6x_6+4x_7 \geq 0\\
&C6: 1+3x_5+6x_6-4x_7 \geq 0
\end{aligned}$$

All the x_i variables are binary: when $x_i = 0$ the corresponding transaction is unsettled and dragged along to the following settlement period, when $x_i = 1$ the corresponding transaction is settled. The model 4 is built with the following code snippet:

```
def create_problem(mu: np.array, sigma: np.array, total: int = 3) -> QuadraticProgram:
    """Solve the quadratic program using docplex."""
    mdl = Model()
    x = [mdl.binary_var("x%s" % i) for i in range(len(sigma))]
    objective = mdl.sum([mu[i] * x[i] for i in range(len(mu))])
    objective -= 2 * mdl.sum(
        [sigma[i, j] * x[i] * x[j] for i in range(len(mu)) for j in range(len(mu))]
    )
    mdl.maximize(objective)
    mdl.add_constraint(5   - 4*x[0] - 3*x[1] - 2*x[2]   >= 0, "cons1")
    mdl.add_constraint(1 + 4*x[0] -3*x[3] -3*x[4]       >= 0, "cons2")
    mdl.add_constraint(2 + 3*x[1]                       >= 0, "cons3")
    mdl.add_constraint(3 + 2*x[2]                       >= 0, "cons4")
    mdl.add_constraint(2 + 3*x[3] - 6*x[5] +  4*x[6]    >= 0, "cons5")
    mdl.add_constraint(1 + 3*x[4] + 6*x[5] -  4*x[6]    >= 0, "cons6")
    qp = from_docplex_mp(mdl)
    return qp
```

The optimization is carried out with the following code:

```
algorithm_globals.random_seed = 12345
algorithm_globals.massive = True

quantum_instance = QuantumInstance(QasmSimulator(method='matrix_product_state'),
                   shots=1)
qaoa_mes = QAOA(quantum_instance=quantum_instance, initial_point=[0.0, 1.0])
exact_mes = NumPyMinimumEigensolver()

Tin3 = time.time()
qaoa = MinimumEigenOptimizer(qaoa_mes)
qaoa_result = qaoa.solve(qubo)
print(qaoa_result)
Tfin3 = time.time() - Tin3
print(f"Total time is :{Tfin3}!")
```

There exist different optimal solutions, for example we found that which envisages the settlement of payments $T2$; $T3$; $T6$; $T7$. This result has been

achieved on This result has been achieved by leveraging the QAOA algorithm available in the Qiskit library. In our example we have employed the all the default argument list for the optimization command. In particular, the integer parameter defining the depth of the ansatz has been set to $p = 1$. We have used only the cost Hamiltonian whereas for the mixer Hamiltonian we have chosen the identity matrix.

References

1. Dorit Aharonov, Wim van Dam, Julia Kempe, Zeph Landau, Seth Lloyd, and Oded Regev. An Introduction to Measurement Based Quantum computation. *SIAM Review*, 50:755–787, 4 2008.
2. Frank et al. Arute. Quantum supremacy using a programmable superconducting processor. *Nature*, pages 1–42, October 2019.
3. Shoelkopf B. and A Smola. Learning with Kernels,. *MIT press*, pages 118–120, 1998.
4. Lee Braine, Daniel J. Egger, Glick Jennifer, and Woerner Stefan. Quantum Algorithms for Mixed Binary Optimization applied to Transaction Settlement. *ArXiv:1910.05788v1-Quantum Physics*, pages 1–8, 10 2019.
5. Frederic T. Chong, Diana Franklin, and Margaret Rose Martonosi. Programming Languages and Compiler design for Realistic Quantum Hhardware. *Nature*, pages 180–187, 9 2017.
6. W. Coss, Andrew, S. Bishop, Lev, Paul Sheldon Sarah, Nation, and Gambetta Jay. Validating quantum computers using randomized model circuits. *Physical Review*, 1, 2019.
7. Avinash Dash, D. Sarmah, B. K. Behera, and P. K. Panigrahi. Exact Search Algorithm to factorize large biprimes and a triprime on ibm quantum computer. *ArXiv:1805.10478v2*, pages 1–13, 2018.
8. Paul Adrien Maurice Dirac. A new Notation for Quantum Mechanics. *Mathematical Proceedings of the Cambridge Philosophical Society*, 35:416–418, 1939.
9. David P. DiVincenzo. The Physical Implementation of Quantum Computation. *Fortschritte der Physik*, 48:771–783, 2000.
10. Daniel Egger, R. Gutierrez, J. C. Mestre, and S. Woerner. Credit Risk Analysis using Quantum computers. *arXiv:1907.03044*, pages 1–8, 2019.
11. Daniel Egger, R. Gutierrez, J. C. Mestre, and S. Woerner. Quantum Computing for Finance: State-of-the-Art and future prospects. *IEEE Transactions on Quantum Engineering*, 1:1–8, 2020.
12. Edward Fahri, Jeffrey Goldstone, and Gutmann Sam. A quantum approximate optimization algorithm. *URL https://arxiv.org/abs/1411.4028.*, 2014.
13. Claudio Gambella and Simonetto Andrea. Multi-block ADMM heuristics for Mixed-Binary Optimization on Classical and Quantum Computers. *ArXiv:2001.02069v2*, 39:1–26, 2 2021.
14. Gartner. https://www.gartner.com/en/documents/3887767/understanding-gartner-s-hype-cycles. *web*, 2020.
15. Oded Goldreich. *P, NP and NP-completeness- the Basics of Computational Complexity*. Cambridge University Press, 2010.
16. Lov K. Grover. Fast quantum mechanical algorithm for database search. *Proceedings of 28th ACM Symposium on Theory of Computing*, pages 212–219, 1996.
17. Lov K. Grover. Quantum Mechanics helps in searching for a needle in a haystack. *Physical Review Letters*, 79:325–328, 1997.
18. Aram W. Harrow, Avinatan Hassidim, and Set Lloyd. Quantum algorithm for linear system of equations. *Physical Review Letters*, 15, 2009.
19. Philippe Jorion. *Value at risk: the new benchmark for managing financial risk*. New York mcGraw-Hill, 2001.
20. Richard Jozsa. An Introduction to Measurement Based Quantum computation. *ArXiv-Quantum Physics*, pages 1–22, 8 2005.

21. Sangil Kwon, Akiyoshi Tomonaga, Gopika Bahi, Devitt Simon, and Jaw-Shen T. Gate-based Superconducting Quantum Computing. *Journal of Applied Physics*, pages 1–51, 1 2021.
22. Ray LaPierre. *Introduction to Quantum Computing*. Springer, 2021.
23. E. Moore, Gordon. Quantum computer 40 years later. *Electronics*, 38:1–42, April 1965.
24. Basel Committee on Banking Supervision. Developments in modelling risk aggregation. *Bank for International Settlements*, October 2010.
25. John Preskill. Quantum Computing in the NISQ era and beyond. *Quantum*, 2:79, August 2018.
26. John Preskill. Quantum computer 40 years later. *ArXiv:1801.00862v3*, 39:1–42, 10 2021.
27. Patrick Rebentrost, Brajesh Gupt, and Thomas R. Bromley. Quantum computational finance: Monte Carlo pricing of financial derivatives. *Physical Review*, 98:1–15, August 2018.
28. Yakov M. Shafransky and Alexander A. Doudkin. An Optimization Algorithm for the clearing of interbank payments. *European Journal of Operational Research*, 171:743–749, 2006.
29. Peter W. Shor. Algorithms for Quantum Computation: Discrete Logarithms and Factoring. *Proceedings 35 Annual Symposium on Foundations of Computer Science*, pages 124–134, 1994.
30. Peter W. Shor. Polynomial-time Algorithms for Prime Factorization and Discrete Logarithms ona Quantum Computer. *SIAM Journal of Computation*, 5:1484–1509, 1997.
31. Hal Varian. High-technology industries and market structure. *Proceedings - Economic Policy Symposium - Jackson Hole*, 1:65–101, 2001.
32. Oldrich Vasicek. Loan portfolio value. *RISK magazine*, pages 160–162, 2002.
33. Jiaxing Wang, Lihua Shen, and Wuyuan Zhou. A Bibliometric Analysis of Quantum Computing literature: mapping and evidences from scopus. *Technology Analysis & Strategic Management*, 33:1347–1363, 2021.
34. Stefan Woerner and D.J. Egger. Quantum risk analysis. *npj Quantum Information*, 5, 2019.