NExt ApplicationS of Quantum Computing

D5.12: Evaluation of quantum algorithms for finance

Document Properties

Contract Number	951821
Contractual Deadline	31-10-2024
Dissemination Level	Public
Nature	Report
Editors	Alberto Manzano, UDC Gonzalo Ferro, CESGA
Authors	María R. Nogueiras, HSBC Gonzalo Ferro, CESGA Alberto Manzano, UDC Andrés Gómez, CESGA Carlos Vázquez, UDC
Reviewers	Mohamed Hibti, EDF Jan Reiner, HQS
Date	28-10-24
Keywords	Quantum Finance, Option Pricing, Amplitude Amplification and Estimation, Quantum Accelerated Monte Carlo, Quantum Machine Learning, Parametric Quantum Circuits Pricing, VaR
Status	Submitted
Release	1.1

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

History of Changes

Release	Date	Author, Organisation	Description of Changes
1.0	01/10/2024	María R. Nogueiras, HSBC Gonzalo Ferro, CESGA Alberto Manzano, UDC Andrés Gómez, CESGA Carlos Vázquez, UDC	First full version
1.1	22/10/2024	María R. Nogueiras, HSBC Gonzalo Ferro, CESGA Alberto Manzano, UDC Andrés Gómez, CESGA Carlos Vázquez, UDC	Revised version after internal reviewer's comments. After removing color text highlighting main changes, it also coincides with the submitted version on 28/10/2024.

Table of Contents

1.Executive Summary

This report presents the advancements in quantum computing techniques for quantitative finance within Use Case 5 (UC5) of the NEASQC project. The research is a collaborative effort among the University of A Coruña (UDC), the Galician Supercomputing Center (CESGA), and the Hong Kong and Shanghai Banking Corporation (HSBC). Each organization brings unique expertise to the project: UDC focuses on theoretical foundations, CESGA on technological implementation, and HSBC on identifying industry-relevant problems.

The research addresses two main areas critical to HSBC: Quantum Accelerated Monte Carlo (QAMC) for option pricing and Quantum Machine Learning (QML) for risk assessment using metrics such as Value at Risk (VaR). These areas are essential for enhancing the efficiency and accuracy of financial modeling and risk management. More specifically, QAMC aims to expedite the option pricing process, while QML provides advanced techniques for risk assessment, offering deeper insights and more robust predictions.

Despite significant progress, the integration of these quantum techniques into industrial applications remains challenging due to current hardware limitations and the early stage of quantum computing technology. The project highlights the need for further development of quantum hardware and algorithms to bridge the technological gap between classical and quantum methods.

Notable advancements include new techniques for QAMC and QML, which are particularly relevant for financial institutions. The project has also produced five research papers and developed a comprehensive software library, QQuantLib, to facilitate further research and experimentation.

In conclusion, while significant progress has been made in developing quantum algorithms for pricing and VaR estimation, these algorithms are not yet enough competitive with classical algorithms on current Noisy Intermediate-Scale Quantum (NISQ) architectures. The primary challenges for moving forward are related to the execution of QAMC and QML algorithms, which require both deeper circuits (more layers of gates) and wider circuits (more qubits) than are currently feasible on available quantum hardware.

2.Introduction

This document focuses on the quantum computing techniques developed for quantitative finance within the context of Use Case 5 (UC5) of the NEASQC project. This research is the result of a collaboration among three principal organizations: the University of A Coruña (UDC), the Galician Supercomputing Center (CESGA), and the Hong Kong and Shanghai Banking Corporation (HSBC). The partnership leverages the complementary expertise of each organization. The University of A Coruña has concentrated on advancing the theoretical foundations and methodologies. The Galician Supercomputing Center has focused on technological implementation and essential support. Meanwhile, HSBC has contributed by identifying and presenting industry-relevant problems related to option pricing and the computation of Value at Risk (VaR), ensuring the practical applicability of the research outcomes.

This research encompasses two main areas, aligned with the primary interests of HSBC. The first area is Quantum Accelerated Monte Carlo (QAMC) for option pricing. The second area is Quantum Machine Learning (QML) for risk assessment, particularly using metrics such as VaR.

QAMC can significantly expedite the option pricing process, allowing for more timely and precise decision-making in volatile markets. This is particularly crucial as classical Monte Carlo methods, while powerful, can be computationally intensive and time-consuming when dealing with high-dimensional problems or when a large number of scenarios need to be simulated. Quantum computing offers the potential to overcome these limitations through its ability to perform certain calculations quadratically faster than classical computers.

Furthermore, as financial institutions like HSBC deal with increasingly complex risk models and regulatory requirements, the need for more sophisticated computational tools becomes apparent. In this scenario QML, presents a promising avenue to address these growing computational demands in ways that classical computers cannot efficiently match.

The document is divided into two parts. The first part concentrates on pricing using Monte Carlo (MC) techniques. Section 3 briefly describes the core concepts of QAMC and its application in the context of pricing. Section 3.2 describes how to perform QAMC in the presence of negative values. This part concludes with Section 3.3, which introduces a new end-to-end workflow, or 'pipeline', for implementing QAMC. This pipeline encompasses the entire process from data preparation to final output generation.

The second part delves into Quantum Machine Learning for risk assessment. Section 4.1 discusses the approximation capabilities of Parametrized Quantum Circuits (PQCs), while Section 4.2 is reserved for recovering the shape of a financial distribution from samples. This represents a key point in the computation of a risk measure like VaR. Finally, the document wraps up with the conclusions.

In order to perform the experiments of the NEASQC project, several libraries have been developed under the QLM platform provided by Atos (Atos, 2016). The primary library used for this report is QQuantLib, a financial applications library available at (Ferro et al., 2024). The documentation of this library can be found under the aforementioned repository. All experiments in part 3 were conducted using this library. Conversely, the experiments in part 4 were performed with the PennyLane library (Bergholm et al., 2018). Additionally, the code for this second part has been migrated to myQLM and can be found in (Ferro et al., 2024).

Finally, we included Appendix A which describes the simplifications made for the experimental evaluation of the QAMC algorithms.

3.Quantum Accelerated Monte Carlo

The problem of derivative pricing can be reduced in many cases to computing an expectation. More specifically, by using mathematical finance tools, mainly martingale properties and Itô's lemma. The price of a derivative contract V_t on an underlying asset S_t can be written as (see, for example (Hull, 1997)):

$$V_t = V(t, S_t) = e^{-r(T-t)}\mathbb{E}^Q[F(S_T)|\mathcal{F}_t], \tag{3.1}$$

where T is the maturity of the contract, $\mathbb{E}^Q$ denotes the expectation under the risk neutral measure Q, r is the constant risk-free interest rate at time t, F defines the payoff of the derivative and $\mathcal{F}_t$ denotes the σ-algebra containing the information until time t. In this way, expression (3.1) indicates that the value of the derivative is the discounted price of the expected value of the payoff, conditioned to the current information of the market.

In the work developed in collaboration with HSBC, we have focused on European vanilla and digital options whose payoff only depends on the value of the asset at maturity, $V_T = F(S_T)$. Thus, if we denote the strike price by K, the corresponding payoffs are (see also Figure 1):

- Vanilla call option: $F(x) = \max(x - K, 0)$.
- Digital call option: $F(x) = 1_{x>K}$.
- Vanilla put option: $F(x) = \max(K - x, 0)$.
- Digital put option: $F(x) = 1_{x<K}$.

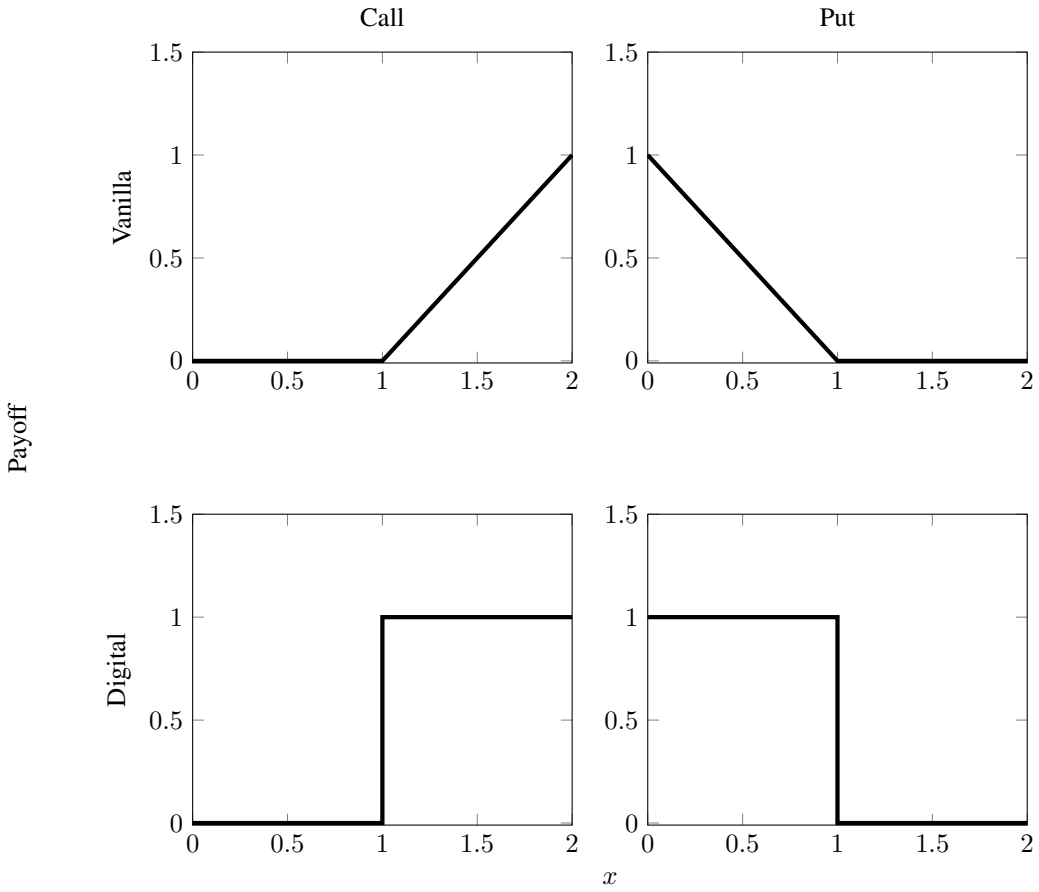

***Figure 1**: Payoff functions for digital and vanilla options with $K = 1$.*

In the previous options, the derivative price is always non-negative. Moreover, in order to illustrate some advantages of one of the proposed quantum algorithms, we have considered linear payoffs:

$$F(x) = x - K,$$

so that the value of the derivative can be also negative (see also Figure 2).

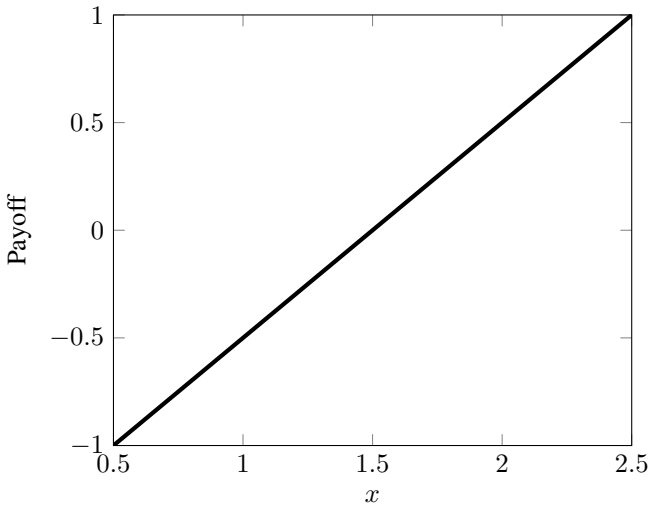

***Figure 2**: Linear payoff with $K = 1.5$.*

The computation of the expectation in Equation (3.1) is usually achived through Classical Monte Carlo (CMC) methods (see (Gómez et al., 2022), for example). However, QAMC is a promising approach that offers a quadratic speedup compared to CMC, making it an attractive option for solving complex financial problems. This technique was first introduced in (Montanaro, 2015). Later, in (Rebentrost et al., 2018) it was proposed for the first time to the task of pricing financial contracts. However, it was not until the publications of (Egger et al., 2020; Stamatopoulos et al., 2020), that QAMC was implemented for pricing and VaR on a real quantum computer.

All of the techniques grouped into the category of QAMC share a common structure or pipeline:

- The construction of an oracle U_S which is able to load the path probability distribution.
- The application of a second unitary U_F which encodes the payoff of the target contract.
- The measurement of a state which encodes the solution of the problem by means of Amplitude Estimation (AE) techniques.

In this section we will focus on an alternative pipeline to perform QAMC. First, in Section 3.1 we describe the standard procedure to perform QAMC (for a more in depth discussion see (Manzano, 2024)). Next, in Section 3.2 we introduce the problem of pricing derivative contracts with negative values under the QAMC framework. Finally, in Section 3.3 we propose an alternative pipeline for QAMC which circumvents some of the burden associated with pricing derivative contracts with negative values.

3.1. Conventional approach to QAMC

The QAMC algorithm begins by creating an oracle U_S which produces samples S according to an underlying probability distribution $p(S_k)$:

$$|S\rangle := U_S\,|0\rangle := \sum_{k=0}^{\mathcal{K}-1} \sqrt{p_S(S_k)}\,|S_k\rangle\,, \tag{3.2}$$

where $\mathcal{K}$ is the number of possible paths and $p_S(S_k)$ is the probability of generating the path S_k (Manzano, Ferro, et al., 2023).

In the case of derivative pricing, the samples represent paths of an underlying asset and the probability distribution is usually modelled through a stochastic differential equation (SDE). The construction of the oracle loading the probability into the quantum state is discussed in more depth in (Manzano, 2024).

The key takeaway is that the computational cost of one execution of the circuit is equivalent to one execution of the classical circuit, *i.e.*, the number of gates needed to sample one path from the classical and the quantum circuit are

"the same", since the classical circuit can always be translated to a quantum one using Toffoli gates (see (Nielsen & Chuang, 2011), for example). However, note that classical and quantum gates are not directly comparable.

Once the state $|S\rangle$ in Equation (3.2) is generated, the next step is to define the operator $U_{\sqrt{F}}$ such that it pushes the square root of the derivatives payoff F into the amplitude. For this reason, we will call this way of encoding *square root encoding*. For this purpose, an additional single qubit register is needed:

$$\begin{aligned} |\sqrt{F}\rangle &= U_{\sqrt{F}} |S\rangle |0\rangle \\ &= \frac{1}{\|\sqrt{F(S)}\|_\infty} \sum_{k=0}^{\mathcal{K}-1} \left(\sqrt{p_S(S_k) F(S_k)} |S_k\rangle |0\rangle + \sqrt{p_S(S_k)\left(1 - F(S_k)\right)} |S_k\rangle |1\rangle \right). \end{aligned} \tag{3.3}$$

Moreover, it is tacitly assumed that the operator $U_{\sqrt{F}}$ can be efficiently implemented. Figure 3 depicts schematically the overall process.

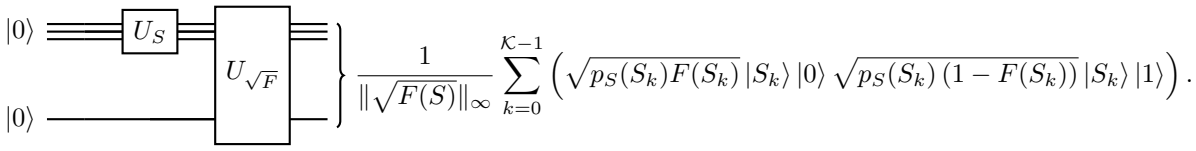

***Figure 3**: Scheme of the generation of the oracle in the square root encoding. The gate U_S corresponds to Equation (3.2). The gate $U_{\sqrt{F}}$ corresponds to Equation (3.3).*

The probability of measuring zero in the rightmost register is given by:

$$P_{|0\rangle} = \frac{1}{\|\sqrt{F(S)}\|_\infty^2} \sum_{k=0}^{\mathcal{K}-1} |p_S(S_k) F(S_k)| \,. \tag{3.4}$$

Hence, getting an estimation $\widetilde{P}_{|0\rangle}$ of $P_{|0\rangle}$ yields an estimation of the expectation in Equation (3.1) except for the normalization constants:

$$\mathbb{E}[F(S_T)|\mathcal{F}_t] = \sum_{k=0}^{\mathcal{K}-1} p_S(S_k) F(S_k) + \epsilon_{\text{S}} \approx \left\| \sqrt{F(S)} \right\|_\infty^2 \widetilde{P}_{|0\rangle} + \epsilon_{\text{S}} + \epsilon_{\text{QAMC}}, \tag{3.5}$$

where $\sum_{k=0}^{\mathcal{K}-1} p_S(S_k) F(S_k)$ is the approximation of the expectation in Equation (3.1), ϵ_{S} is the error from approximating the expectation, $\widetilde{P}_{|0\rangle}$ is an estimation of the probability of obtaining zero in the last register and ϵ_{QAMC} is the sampling error defined as,

$$\epsilon_{\text{QAMC}} := \left| \sum_{k=0}^{\mathcal{K}-1} p_S(S_k) F(S_k) - \left\| \sqrt{F(S)} \right\|_\infty^2 \widetilde{P}_{|0\rangle} \right|. \tag{3.6}$$

By using amplitude estimation techniques, we know that the sampling error ϵ_{QAMC} is of order (Brassard et al., 2002):

$$\epsilon_{\text{QAMC}} \sim \frac{1}{N_Q}, \tag{3.7}$$

with N_Q being the number of calls to the oracle defined by Equation (3.3). In the sampling error scales as:

$$\epsilon_{\text{CMC}} \sim \frac{1}{\sqrt{N_C}}. \tag{3.8}$$

Since we argued that a call to the quantum oracle is equivalent to the generation of a sample in the classical case, we say that there is a quadratic speedup.

To wrap up this section, in Figure 4 we show the results of the QAMC for different payoffs when the modified Iterative Quantum Amplitude Estimation (mIQAE) algorithm is used to estimate $P_{|0\rangle}$. Further details on the implementation can be found in the Appendix A.

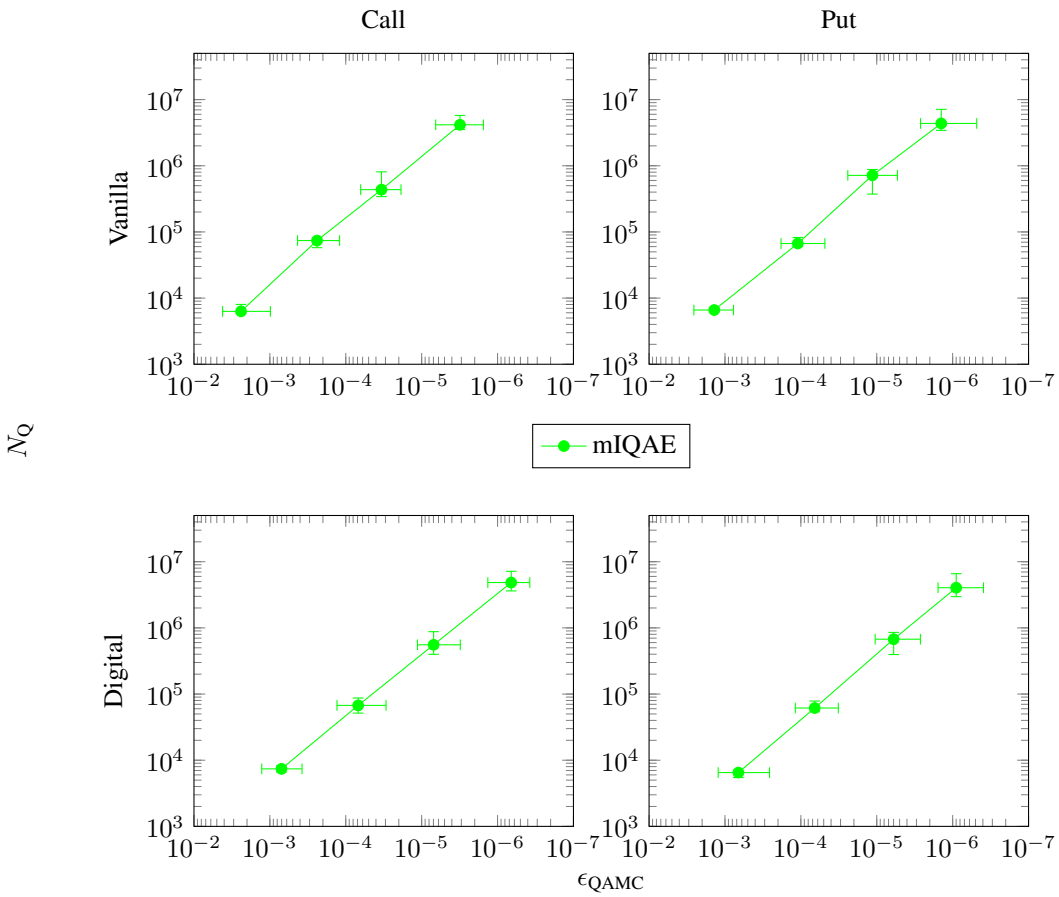

***Figure 4**: Absolute error between the QAMC algorithm and the discretized expectation versus the respective number of calls to the oracle for different precisions ϵ. The dots represent the medians and the error bars the 25 and 75 percentiles. Each of the panels corresponds to the payoff of different options. The experiments have been performed using the square root encoding.*

3.2. QAMC in the presence of negative expectations

It is important to note that, for derivatives whose payoffs can become negative, the naive use of QAMC will not yield correct price approximations. In order to illustrate this, suppose that there is a payoff of the form:

$$F(S_T) = S_T - K, \tag{3.9}$$

with T being the maturity of the contract, S_T the price of the underlying asset at maturity and K the strike price of the contract (see (Gómez et al., 2022) for details). Figure 5 shows the results for the the square root encoding combined with the mIQAE for a naive implementation of QAMC. It illustrates that there is no convergence to the correct value because of the presence of the absolute value in Equation (3.4).

In order to avoid the errors introduced by the presence of the absolute values in the QAMC, whenever we have a payoff that is potentially negative we must divide our problem into two distinct problems. On the one hand, we must define the positive part of our target function:

$$F_+(S_T) = \max(F(S_T), 0). \tag{3.10}$$

On the other hand, we define the negative part of our target function:

$$F_-(S_T) = |\min(F(S_T), 0)|\,. \tag{3.11}$$

Therefore, we can express our payoff as a linear combination of the positive and negative parts:

$$F(S_T) = F_+(S_T) - F_-(S_T). \tag{3.12}$$

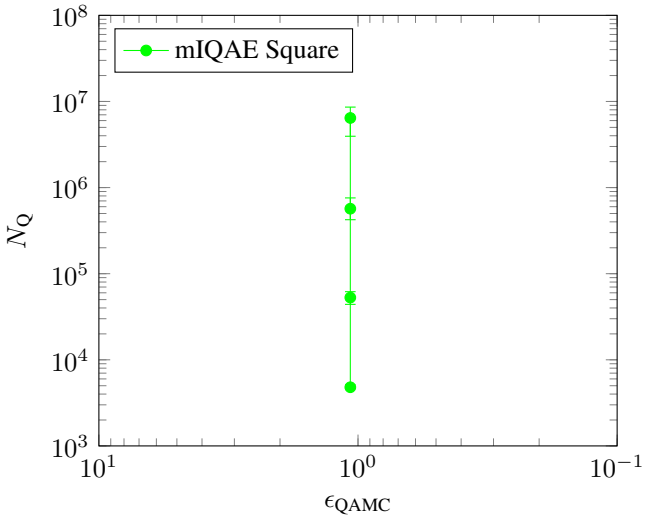

***Figure 5**: Absolute error between the QAMC algorithm and the discretized expectation of an option with a payoff $(S_T - K)$ with $K = 1.5S_t$ versus the number of calls to the oracle for different values of precision ϵ. The dots represent the medians and the error bars the 25 and 75 percentiles. The experiments have been performed using the square root encoding.*

In terms of estimation we need to perform a separate estimation of both the positive and negative parts using the conventional approach to QAMC described in Section 3.1 and then combine the results. The only issue that we have to take into consideration when combining the positive and negative estimations is that the errors add up together,

$$\epsilon_{\text{QAMC}} = \epsilon^{+}_{\text{QAMC}} + \epsilon^{-}_{\text{QAMC}}, \tag{3.13}$$

so that, in order to get a total error of ϵ_{QAMC} we usually take:

$$\epsilon^{+}_{\text{QAMC}} = \frac{\epsilon_{\text{QAMC}}}{2}, \qquad \epsilon^{-}_{\text{QAMC}} = \frac{\epsilon_{\text{QAMC}}}{2}. \tag{3.14}$$

Applying this decomposition, the mIQAE provides the results in Figure 6.

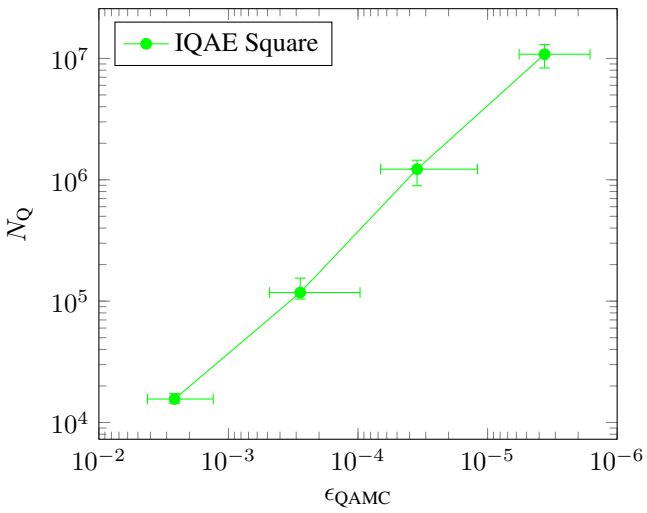

***Figure 6**: Absolute error between the QAMC algorithm and the discretized expectation of an option with a payoff $(S_T - K)$ with $K = 1.5S_t$ versus the number of calls to the oracle for different values of precision ϵ. The dots represent the medians and the error bars the 25 and 75 percentiles. The experiments have been performed using the square root encoding separating the positive and negative parts of the payoff.*

3.3. An alternative pipeline for QAMC

In this section, we develop a new strategy which does not require the user to separate the problem into two. On the one hand, in Section 3.3.1 a new encoding is proposed. On the other hand, in Section 3.3.2 a different amplitude estimation technique is employed. The combination of both produces an alternative pipeline to the standard QAMC.

3.3.1. Direct Encoding

The direct encoding algorithm starts from the same initial state $|S\rangle$:

$$|S\rangle = U_S\,|0\rangle = \sum_{k=0}^{\mathcal{K}-1} \sqrt{p_S(S_k)}\,|S_k\rangle\,, \tag{3.15}$$

where $\mathcal{K}$ is again the number of possible paths defined by the given discretization. The next step is to define the operator U_F such that it pushes the payoff without squared roots into the amplitude. For this purpose, an additional single qubit register is needed, so that:

$$\begin{aligned} |F\rangle &= U_F\,|S\rangle\,|0\rangle \\ &= \frac{1}{\|F(S)\|_\infty} \sum_{k=0}^{\mathcal{K}-1} \left(\sqrt{p_S(S_k)}F(S_k)\,|S_k\rangle\,|0\rangle + \sqrt{p_S(S_k)}\,(1-F(S_k))\,|S_k\rangle\,|1\rangle \right). \end{aligned} \tag{3.16}$$

Next, we apply the inverse of the U_S unitary on the state $|F\rangle$ (see Figure 7), thus getting:

$$U_S^\dagger\,|F\rangle = \frac{1}{\|F(S)\|_\infty} \sum_{k=0}^{\mathcal{K}-1} \left(p_S(S_k)F(S_k)\,|0\rangle\,|0\rangle + \cdots \right). \tag{3.17}$$

The square root probability of measuring the eigenstate zero is:

$$\sqrt{P_{\overline{|0\rangle}}} = \left| \langle 0|\, U_S^\dagger U_F U_S\,|0\rangle \right| = \frac{1}{\|F(S)\|_\infty} \left| \sum_{k=0}^{\mathcal{K}-1} p_S(S_k)F(S_k) \right|. \tag{3.18}$$

Note the difference between $P_{|0\rangle}$ from the square root encoding in Equation (3.5) and $P_{\overline{|0\rangle}}$ from the direct encoding in Equation (3.18). The former one refers to the probability of measuring zero in the last register when the unitary $U_{\sqrt{F}}U_S$ is applied. The latter refers to the probability of measuring the eigenstate zero when the unitary $U_S^\dagger U_F U_S$ is applied. Finally, we apply an AE algorithm to obtain an estimate $\widetilde{P}_{\overline{|0\rangle}}$ of $P_{\overline{|0\rangle}}$, thus getting an estimation of the

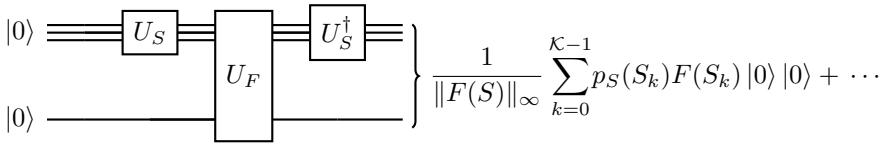

***Figure 7**: Scheme of the generation of the oracle in the direct encoding. The gate U_S corresponds to Equation (3.2). The gate U_F corresponds to Equation (3.16).*

expectation in Equation (3.1):

$$\mathbb{E}[F(S_T)|\mathcal{F}_t] = \sum_{k=0}^{\mathcal{K}-1} p_S(S_k)F(S_k) + \epsilon_S \approx \|F(S)\|_\infty \sqrt{\widetilde{P}_{\overline{|0\rangle}}} + \epsilon_S + \epsilon_{\text{QAMC}}, \tag{3.19}$$

where $\sum_{k=0}^{\mathcal{K}-1} p_S(S_k)F(S_k)$ is the approximation of the expectation in Equation (3.1), ϵ_S is the error from approximating the expectation, $\widetilde{P}_{|0\rangle}$ is an estimation of the probability of obtaining zero in the last register and ϵ_{QAMC} is the sampling error given by,

$$\epsilon_{\text{QAMC}} := \left| \sum_{k=0}^{\mathcal{K}-1} p_S(S_k)F(S_k) - \|F(S)\|_\infty \sqrt{\widetilde{P}_{\overline{|0\rangle}}} \right|. \tag{3.20}$$

This procedure allows pricing options with negative payoffs when the expected value is positive. However, the presence of the outer absolute value in Equation (3.18) still prevents from a correct estimation when negative expectations arise.

In Figure 8 we show the results obtained for the same payoffs as in Figure 4 with both the direct and the square root encoding. As we can see from the figure, the impact of using one or the other encoding is minimal in practice.

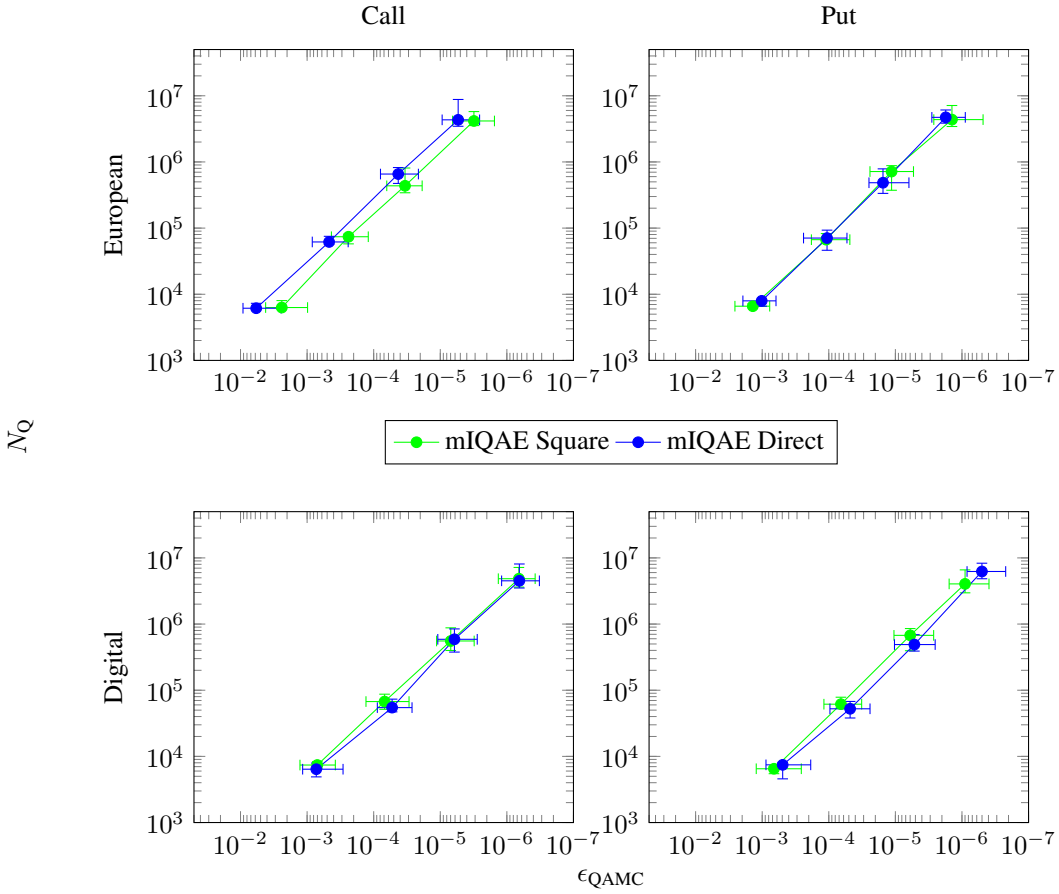

***Figure 8**: Absolute error between the QAMC algorithm and the discretized expectation versus the respective number of calls to the oracle for different precisions ϵ. The dots represent the medians and the error bars the 25 and 75 percentiles. Each of the panels corresponds to the payoff of a different option. The experiments have been performed using the square root and direct encodings.*

3.3.2. Amplitude Estimation: mRQAE

In the previous section it was discussed that the discretized expectation can be estimated through the probability of measuring the eigenstate zero of U_{DE}:

$$\sqrt{P_{|0\rangle}} = \frac{1}{\|F\|_\infty} \left| \sum_{k=0}^{\mathcal{K}-1} p_S(S_k) F(S_k) \right| .$$

Thus, this partially solves the initial problem. Instead of obtaining the sum of absolute values, something proportional to the absolute value of the sum is returned. Hence, in a situation where the sign of the expectation is of interest, an additional mechanism to overcome this issue is needed. In fact, this is usually the case in financial applications, where the sign makes the difference between a profit and a loss.

For this case, we introduce the modified real quantum amplitude estimation (mRQAE) algorithm. The mRQAE is a modified version of the real quantum amplitude estimation (RQAE) (see (Manzano, Musso, & Leitao, 2023)). The main feature of both algorithms is that they are able to read out the size and the sign of the target amplitude. They

internally work performing several iterations where, in each iteration they use the Grover amplification algorithm to incrementally amplify the probability of obtaining the target quantum state. The main difference between them is that the number of calls to the amplified state N_i, the confidence on each iteration γ_i and the required precision on each iteration ϵ_i^p are chosen in a different manner. In turn, this makes the mRQAE asymptotically more efficient. More precisely, in Table 1 we show the performance of the mRQAE and the RQAE measured in terms of the number of calls to the oracle N_Q for a given precision ϵ and confidence γ along with the performance of other popular amplitude estimation algorithms in the literature including the aforementioned mIQAE. In Algorithm 1 there is a schematic description of the code for the mRQAE. For a more thorough revision of the properties of the method we refer the reader to (Manzano, 2024).

Algorithm 1 mRQAE pseudocode.

Input:
- ϵ // Required precision
- γ // $1-\gamma$ is the confidence level
- q // Amplification policy
- $\mathcal{A}$ // Oracle

Output:
- a // Estimated amplitude with sign

Algorithm:

// Define relevant parameters

Set $\epsilon^p(q,\infty) = \frac{1}{2}\sin^2\left(\frac{\pi}{4q}\right)$

Set $T = \log_q\left(q^2\frac{2\arcsin\left(\sqrt{2\epsilon^p(q,\infty)}\right)}{\arcsin(2\epsilon)}\right)$

Set $k^{\max} = \left\lceil \frac{\arcsin\left(\sqrt{2\epsilon^p(q,\infty)}\right)}{\arcsin(2\epsilon)} - \frac{1}{2}\right\rceil$

Set $\epsilon^p(q,0) = \frac{1}{2}\sin\left(\frac{\pi}{2(q+2)}\right)$

$i = 1$ // First Iteration

Set $\gamma_i = \frac{\gamma}{2}\frac{q-1}{q}\frac{1}{2k^{\max}+1}$ // Confidence for each iteration

Set $N_i = \left\lceil \frac{1}{2(\epsilon^p(q,0))^2}\log\left(\frac{2}{\gamma_i}\right)\right\rceil$ // Number of shots

Set $\epsilon_i^p = \sqrt{\frac{1}{2N_1}\log\left(\frac{2}{\gamma_i}\right)}$

Set $b = 0.5$ // Shift

Measure p_{sum} and p_{diff}

$a^{\max} = \min\left(\frac{\hat{p}_{\text{sum}} - \hat{p}_{\text{diff}}}{4b} + \frac{\epsilon_i^p}{|2b|}, 1\right)$

$a^{\min} = \max\left(\frac{\hat{p}_{\text{sum}} - \hat{p}_{\text{diff}}}{4b} - \frac{\epsilon_i^p}{|2b|}, -1\right)$

$a = \frac{a^{\max}+a^{\min}}{2}$

$\epsilon^a = \frac{a^{\max}-a^{\min}}{2}$ // Following Iterations

while $\epsilon^a > \epsilon$ **do**

$i = i+1$

Set $b = -a^{\min}$ // Shift

Set $k = \left\lfloor \frac{\pi}{4\arcsin(2\epsilon^a)} - \frac{1}{2}\right\rfloor$ // Number of amplifications

$k = \min(k, k^{\max})$

Set $\epsilon^p(q,k) = \frac{1}{2}\sin^2\left(\frac{\pi}{4\left(q+\frac{2}{2k+1}\right)}\right)$

Set $\gamma_i = \frac{\gamma}{2}\frac{q-1}{q}\frac{2k+1}{2k^{\max}+1}$ // Confidence

Set $N_i = \left\lceil \frac{1}{2(\epsilon^p(q,k))^2}\log\left(\frac{2}{\gamma_i}\right)\right\rceil$ // Number of shots

Set $\epsilon_i^p = \sqrt{\frac{1}{2N_i}\log\left(\frac{2}{\gamma_i}\right)}$

Measure p // Shifted probability with k amplifications

$p^{\max} = \min(p+\epsilon_i^p, 1)$

$p^{\min} = \max(p-\epsilon_i^p, 0)$

$\theta^{\max} = \frac{\arcsin(\sqrt{p^{\max}})}{2k+1}$

$\theta^{\min} = \frac{\arcsin(\sqrt{p^{\min}})}{2k+1}$

$a^{\max} = \sin\left(\theta^{\max}\right) - b$

$a^{\min} = \sin\left(\theta^{\min}\right) - b$

$a = \frac{a^{\max}+a^{\min}}{2}$

$\epsilon^a = \frac{a^{\max}-a^{\min}}{2}$

end while

return a

In Figure 9, an example where the price of the derivative becomes negative is shown. As it is shown, mRQAE is able to recover the true price without requiring additional mechanisms. Moreover, it is competitive with the mIQAE which is currently considered one of the most efficient algorithms in the literature in terms of number of calls to the oracle.

In Figure 10, we show that the mRQAE obtains a reasonable performance compared with the mIQAE when we use it for contracts with positive payoff.

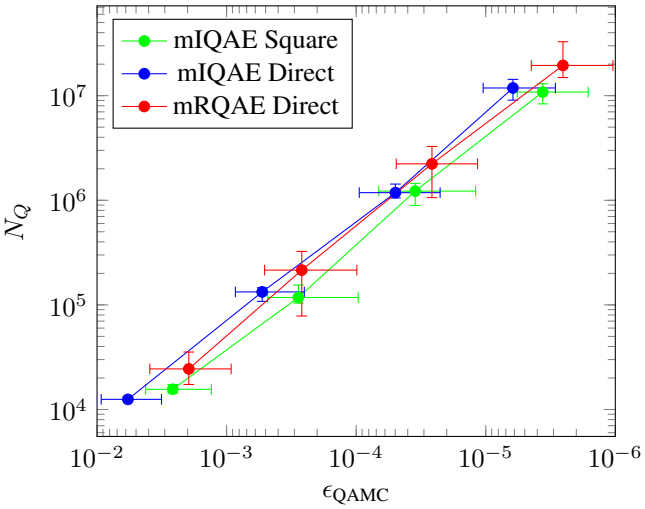

***Figure 9**: Absolute error between the QAMC algorithm and the discretized expectation of an option with a payoff $(S_T - K)$ with $K = 1.5S_t$ versus number of calls to the oracle for different values of precision ϵ. The dots represent the medians and the error bars the 25 and 75 percentiles. For the mRQAE we have used the direct encoding and for the mIQAE we have applied both techniques. Moreover, in the case of the mIQAE we have separated the negative and positive parts of the payoff for a correct pricing.*

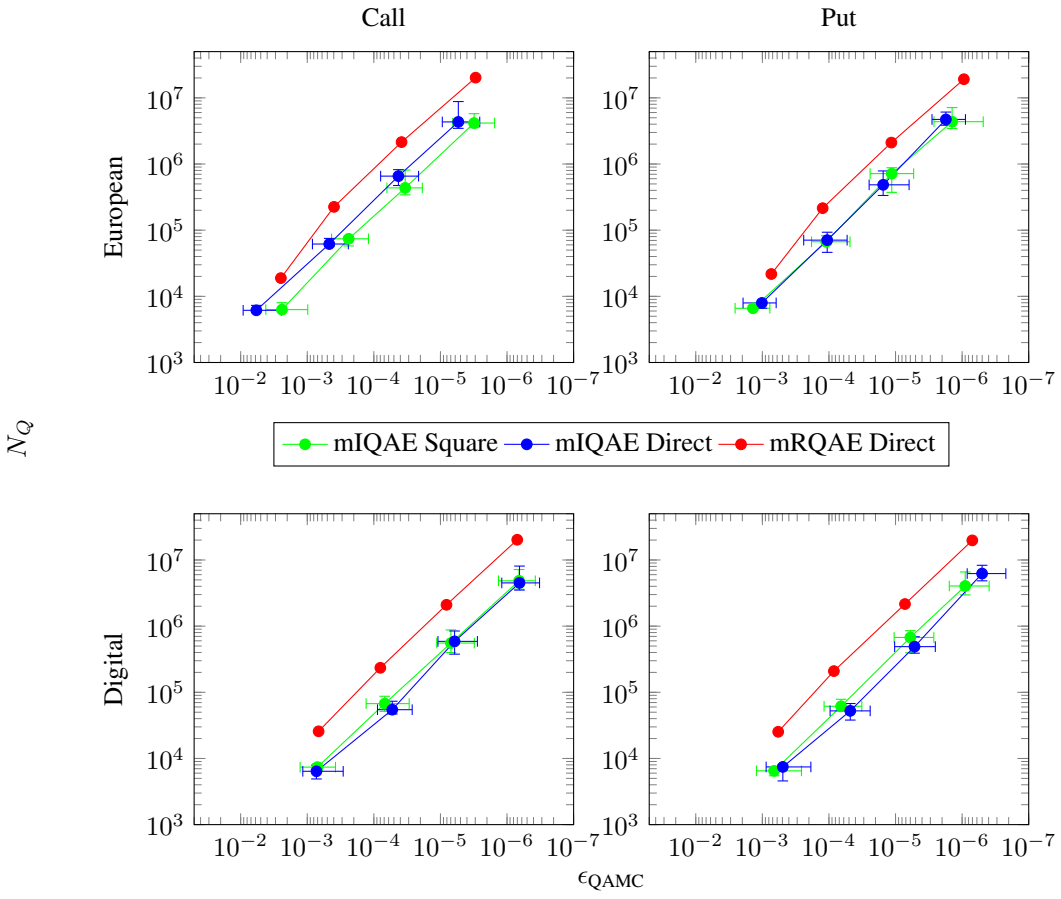

***Figure 10**: Absolute error between the QAMC algorithm and the discretized expectation versus the respective number of calls to the oracle for different precisions ϵ. The dots represent the medians and the error bars the 25 and 75 percentiles. Each of the panels corresponds the payoff of a different option. For the mRQAE we have used the direct encoding and for the mIQAE we have applied both techniques.*

Algorithm	Performance
Monte Carlo	$N_Q^{\text{MC}} = \mathcal{O}\left(\frac{1}{\epsilon^2}\right)$
QPE(Brassard et al., 2002)	$N_Q^{\text{QPE}} = \mathcal{O}\left(\frac{1}{\epsilon}\right)$
MLAE-LIS(Suzuki et al., 2020)	$N_Q^{\text{LIS}} = \mathcal{O}\left(\frac{1}{\epsilon^{\frac{4}{3}}}\right)$
MLAE-EIS(Suzuki et al., 2020)	$N_Q^{\text{EIS}} = \mathcal{O}\left(\frac{1}{\epsilon}\right)$
PLAE(Giurgica-Tiron et al., 2022)	$N_{\mathcal{A}}^{\text{PLAE}} = \mathcal{O}\left(\frac{1}{\epsilon^{1+\beta}}\right), d = \mathcal{O}\left(\frac{1}{\epsilon^{1-\beta}}\right)$
Improved MLAE(Callison & Browne, 2022)	$N_Q^{\text{impMLAE}} = \mathcal{O}\left(\frac{1}{\epsilon}\frac{1}{d}\log(\frac{1}{\gamma})\right), d = 2^{q-2}$
IQAE (Grinko et al., 2021)	$N_Q^{\text{IQAE}} < \frac{50}{\epsilon}\log\left(\frac{2}{\gamma}\log_2\frac{\pi}{4\epsilon}\right)$
mIQAE(Fukuzawa et al., 2023)	$N_Q^{\text{mIQAE}} < \frac{123}{\epsilon}\log\frac{6}{\gamma}$
QCoin (Abrams & Williams, 1999)	$N_Q^{\text{QCoin}} = \mathcal{O}\left(\frac{1}{a}\frac{1}{\epsilon}\log\frac{1}{\gamma}\right), k \geq 2, 1 \geq q \geq (k-1)$
QoPrime (Giurgica-Tiron et al., 2022)	$N_Q^{\text{QoPrime}} < C\lceil\frac{k}{q}\rceil\frac{1}{\epsilon^{1+q/k}}\log\left(\frac{4}{\gamma}\lceil\frac{k}{q}\rceil\right), d = \mathcal{O}\left(\frac{1}{\epsilon^{1-q/k}}\right)$
FasterAE (Nakaji, 2020)	$N_Q^{\text{fasterAE}} < \frac{4.1\cdot 10^3}{\epsilon}\log\left(\frac{4}{\gamma}\log_2\left(\frac{2\pi}{3\epsilon}\right)\right)$
AdaptiveAE (Zhao et al., 2022)	$N_Q^{\text{adaptiveAE}} < \mathcal{O}\left(\frac{1}{\epsilon}\log\left(\frac{\pi^2(T+1)}{3\gamma}\right)\right), T = \lceil\frac{\log\frac{\pi}{K\epsilon}}{\log K}\rceil$
RQAE (Manzano, Musso, & Leitao, 2023)	$N_Q^{\text{RQAE}} < \frac{C_1(q)}{\epsilon_a}\log\left[\frac{3.3}{\gamma}\log_q\left(\frac{C_2(q)}{\epsilon}\right)\right]$
mRQAE (Manzano, 2024)	$N_Q^{\text{MRQAE}} < \frac{C_1(q)}{\epsilon}\log\left[\frac{C_2(q)}{\gamma}\right]$

***Table 1**: Performance of different amplitude estimation algorithms. N_Q denotes the number of calls to the oracle, ϵ is the target precision and $1-\gamma$ is the confidence level. Other parameters appearing in the table are related to each specific algorithm. For a full description of their meaning the reader is referred to the associated references. The $\sim$ symbol indicates that the algorithm has an asymptotic behaviour, while the $<$ indicates that the performance is proved rigorously.*

4.Quantum machine learning for risk assessment

Machine learning has gained significant attention in recent years for its practical applications and transformative impact in various fields. As a consequence, there has been a rising interest in exploring the use of quantum circuits as machine learning models, capitalizing on the advancements in both fields to unlock new possibilities and potential breakthroughs. Among the various possibilities for leveraging quantum circuits in machine learning, our particular focus lies on parametrized quantum circuits (PQC). These quantum circuits consist of both fixed and adjustable (hence 'parametrized') gates. When used for a learning task, a classical optimiser updates the parameters of the PQC in order to minimize a cost function depending on measurement results from this quantum circuit (see Figure 11).

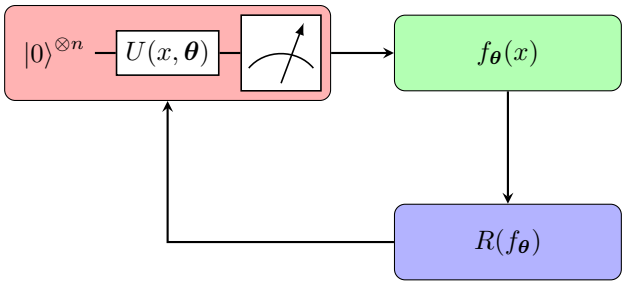

***Figure 11**: Sketch of a hybrid variational algorithm. $U(x,\theta)$ represents a quantum circuit that takes x as input and with variational parameters θ, $f_\theta(x)$ is the expected value of some observable and $R(f_\theta)$ is the expected loss that we want to minimize.*

More specifically, we use the same structure for the PQCs as in (Schuld et al., 2021), where they considered a quantum machine learning model of the form

$$f_\theta(\boldsymbol{x}) = \langle 0| U^\dagger(\boldsymbol{x};\boldsymbol{\theta}) M U(\boldsymbol{x};\boldsymbol{\theta}) |0\rangle \,, \tag{4.1}$$

where M is an observable, $U(\boldsymbol{x};\boldsymbol{\theta})$ is a quantum circuit modelled as a unitary that depends on inputs $\boldsymbol{x} = (x_0, x_1, ..., x_N)$ and the variational parameters $\boldsymbol{\theta} = (\theta_0, \theta_1, ..., \theta_T)$. Note that in Equation (4.1) we understand that the quantum model $f_\theta(\boldsymbol{x})$ is obtained as the expectation value of the observable with respect to the state prepared via the parametrized circuit from the initial state $|0\rangle$. Throughout the report we will refer to the PQC as the one approximating the functions to make the text more fluent, although technically it is the expectation value of the PQC that approximates the function.

The quantum circuit consists of L layers, each one composed by a trainable circuit block $W_i(\boldsymbol{\theta}), i \in \{1, ..., L\}$ and a data encoding block $S(\boldsymbol{x})$ as shown in Figure 12. Such structure is called a re-uploading structure (Pérez-Salinas et al., 2020), since the input is uploaded multiple times along the circuit.

The specific application that we are interested in is financial risk. When a financial institution deals with risk it typically needs to estimate (either implicit or explicitly) the "shape" of an underlying distribution from samples. This estimation is specially delicate in the tails of the distributions, since the tails are usually very difficult to estimate with precision and are connected to unlikely events which can potentially shatter the financial stability of a firm. For this reason, there exist very popular risk metrics such as Value at Risk (VaR) which measures the potential loss in case of an unlikely event. This risk metric is straightforward to compute once that we have obtained the shape of the financial distribution, since we only need to compute a quantile of the distribution. The application of differential machine learning for recovering the shape of the tails can have a meaningful impact since it forces our estimates of the distribution to converge to the distribution pointwise and not just on average. However, in order for differential machine learning to work with samples instead of labels some adjustments need to be done.

This part of the report is organized in two main blocks. First, in Section 4.1 we discuss the impact of normalization in PQCs. Second, in Section 4.2 we apply a new loss function to the problem of recovering the shape of a financial distribution from samples.

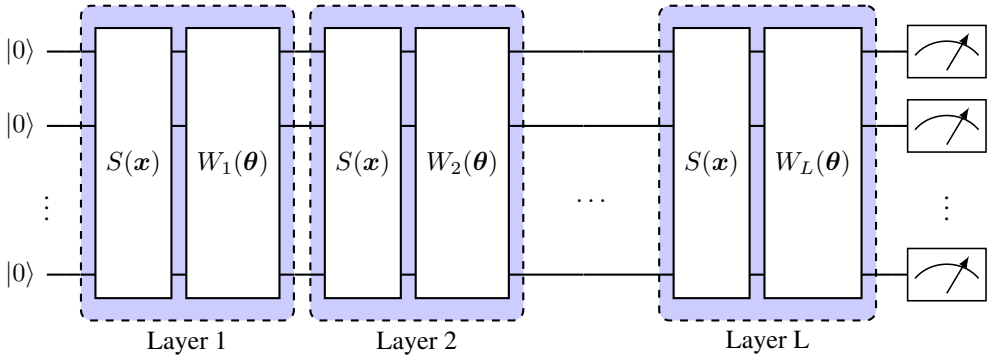

***Figure 12**: Parametrized quantum circuit that can be written as a generalized trigonometric series as in (4.1). It consists of L layers, each layer is composed by a trainable circuit block $W_i(\boldsymbol{\theta}), i \in \{1, ..., L\}$ and a data encoding block $S(\boldsymbol{x})$. The data encoding blocks $S(\boldsymbol{x})$ are identical for all layers, they are typically implemented using Pauli rotations. The blocks $W_i(\boldsymbol{\theta})$ can be built from local rotation gates and $CNOT$ gates.*

4.1. The effect of normalization of PQCs

In this section we illustrate the practical implications of data normalization with the approximation of the function:

$$f^*(x) = \frac{x}{2\pi}, \quad x \in [-\pi, \pi], \tag{4.2}$$

by the PQC in Figure 13.

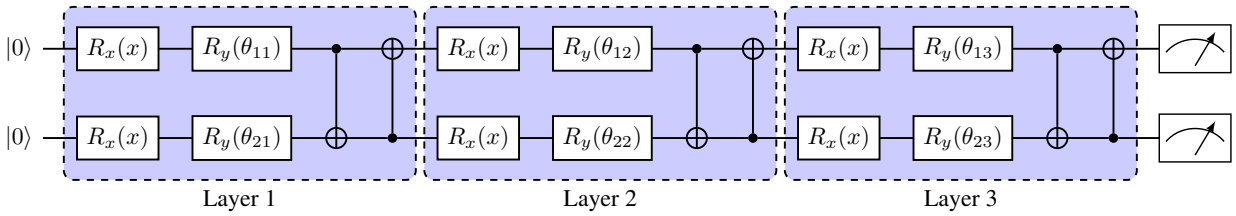

***Figure 13**: Architecture $U(x, \boldsymbol{\theta})$ used in the experiments of Section 1.3. The parameters θ_{ij} are variational parameters.*

We conduct a numerical experiment to compare the performance of our PQC under different normalizations and show the obtained results in Figure 14. In this framework, we are given a data set $\mathcal{S}$ of $I = 10$ pairs of independent samples (10 for the labels plus 10 for the derivative values when they are present) uniformly distributed along the domain for the training phase. Each pair is composed of an input x defined in a domain $\mathcal{X} \in [-\pi, \pi]$ and and output y defined in a co-domain $\mathcal{Y} \in [-1/2, 1/2]$ sampled according to a uniform probability distribution U:

$$\mathcal{S} = \{(x_i, f^*(x_i)) \in \mathcal{Z} = \mathcal{X} \times \mathcal{Y} \sim U(x) : \forall i \in \{1, ..., I\}\}. \tag{4.3}$$

All simulations have been performed using 10 points. Each experiment has been repeated 100 times and we depict the 25, 50 and 75 percentiles in colored solid lines in Figure 14. The legends denote the result of the PQCs as $f_\bullet(\cdot)$, where the subscript denotes under which loss function we have done the training and, in the parentheses, we indicate which normalisation we have chosen.

In this case, the loss function that we have chosen is the mean squared error (here denoted by L^2):

$$R^{\mathcal{S}}_{L^2}(f) = \frac{1}{I}\sum_{i=0}^{I-1}(f^*(x_i) - f(x_i))^2, \tag{4.4}$$

and we normalised the data to lie in the domains $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$, $[-\pi, \pi]$ and $[-2\pi, 2\pi]$, respectively. The experiments h

As illustrated in Figure 14, when we normalise our data to lie in the range $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$ we get the best results, the

second best for the interval $[-\pi, \pi]$ and the worst one with the interval $[-2\pi, 2\pi]$. This behaviour remains even when increasing the size of the circuit and the number of given points.

This results are a consequence of the approximation capabilities of each circuits depending on the normalization.

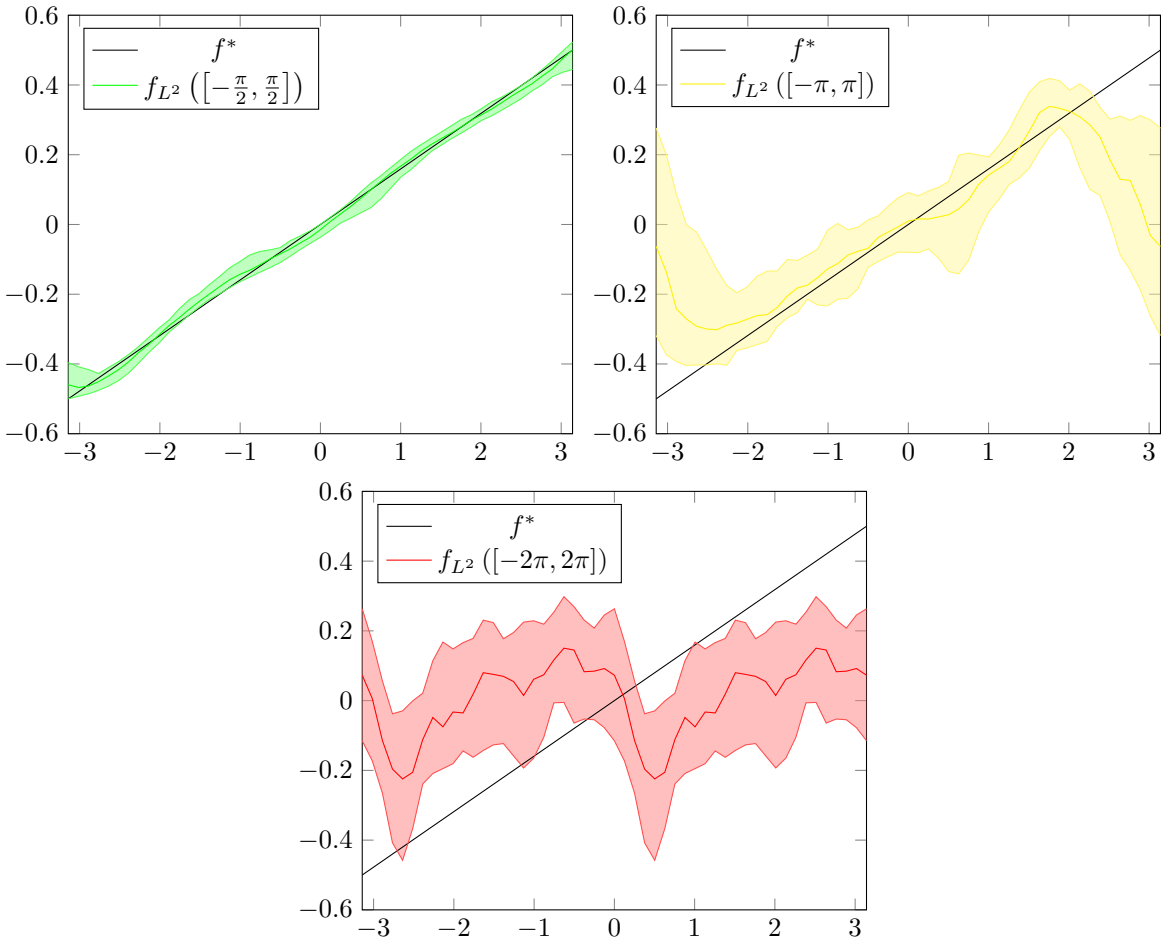

***Figure 14**: In this picture we have trained the PQC of Figure 13 to approximate the function $f^*(x) = \frac{x}{2\pi}$. We have used 10 training points, the mean squared error loss function and 100 epochs with the Adam optimizer. The experiments have been repeated 100 times. In the top left panel we have normalised the data to lie in the interval $[-\frac{\pi}{2}, \frac{\pi}{2}]$. In the top right panel we have normalised the data to lie in the interval $[-\pi, \pi]$. In the bottom central panel we have normalised the data to lie in the interval $[-2\pi, 2\pi]$.*

More specifically, Theorem 5.2.2 from (Manzano, 2024) proofs that under the normalization $[-\frac{\pi}{2}, \frac{\pi}{2}]$ we are able to approximate any function in the sense of the C^0 space. In layman terms this means that we are able to arbitrarily reduce the maximum error as we increase the expressivity of the circuit and the number of samples. On the contrary, Theorem 5.2.1 from (Manzano, 2024) proofs that under the normalization $[-\pi, \pi]$ we are only able to approximate any function in the sense of the L^2 space. In layman terms this means that we are able to reduce the "average" error as we increase the expressivity of the circuit and the number of samples but we are not able to reduce the maximum error. Contrary to the intuition, if we increase the expressivity and the number of samples the results on the boundaries become poorer and poorer. In other terms, since we are only reducing the error on average we see a tradeoff: we improve the results in the bulk of the domain while we deteriorate it in regions which become narrower. Although the results in certain regions, since those regions become smaller at a faster pace we obtain positive net effect, *i.e.*, the error is reduced.

However, the fact that we **can** reduce the maximum error does not mean that the maximum error **is** reduced. Apart from the approximation capabilities of our circuit there are more pieces into play. Namely, the election of the loss function and the minimization routine. Here we don't discuss the minimization routine, but, in the next section we discuss further the impact of the loss function.

4.2. Learning a financial distribution from samples with PQCs

As we have seen, proper normalization of our data greatly improves the expressivity of our PQC, which potentially affects the overall performance of our PQC. However, the minimization problem is made of various pieces, each of which plays a relevant role in their own terms and, if we want to boost the overall performance of the algorithm, we need to work on each of them. The piece that we will focus hereon is the loss function. Our main goal is reducing not only the average error, but also the maximum one.

For this purpose, in the work (Huge & Savine, 2020) they introduce the concept of differential machine learning (DML). The idea is to compute a loss function depending not only in the target function but also on the derivatives of the target function with respect to the inputs. In the same work they showed that this trick can greatly improve the performance of our machine learning models in practice. Moreover, in (Manzano, 2024) they theoretically proof that the minimization of such loss function produces a minimization of the maximum error, with the advantage of being differentiable. However, there is a big caveat: when do we have this information?

In this section, we present an extension of the idea of differential machine learning to the task of learning a Cumulative Distribution Function (CDF) and its derivative, the Probability Density Function (PDF), from samples.

When we work in the context of probability distributions, instead of working with labeled samples we work with samples from the probability distribution, *i.e.*, we have a dataset of the form:

$$\mathcal{S}_{\mathcal{X}} = \{(\boldsymbol{x}_i) \in \mathcal{X} : \boldsymbol{x}_i \sim F^*,\, i \in \{0, ..., I-1\}\}, \tag{4.5}$$

where I is the number of samples, F^* denotes the distribution function and $\boldsymbol{x}_i = (x_i^{(1)}, ..., x_i^{(n)})$.

In this context, we will define an empirical risk depending on both, the distribution function F^* and the density function f^*. This loss function will consist of two parts.

The first part of the empirical risk $R_{L^2}^{\mathcal{S}_{\mathcal{X}}}(F)$ only depends on the distribution function and is defined as:

$$R_{L^2}^{\mathcal{S}_{\mathcal{X}}}(F) = \frac{1}{I}\sum_{i=0}^{I-1}\left(F^*_{\text{emp}}(\boldsymbol{x}_i) - F(\boldsymbol{x}_i)\right)^2, \tag{4.6}$$

where the empirical distribution function[1] $F^*_{\text{emp}}(\boldsymbol{x})$ is defined as follows:

$$F^*_{\text{emp}}(\boldsymbol{x}) = \frac{1}{I}\sum_{i=0}^{I-1}\mathbf{1}_{\boldsymbol{x^i}\leq\boldsymbol{x}}. \tag{4.7}$$

The second part of the empirical risk $R_{\overline{L}^2}^{\mathcal{S}_{\mathcal{X}}}(f)$ only depends on the density function and is defined as:

$$R_{\overline{L}^2}^{\mathcal{S}_{\mathcal{X}}}(f) = -\frac{2}{I}\sum_{i=0}^{I-1} f(\boldsymbol{x}_i) + Q(f^2), \tag{4.8}$$

where $Q(f^2)$ denotes the integral of f^2 over the domain.

Combining the empirical risk for the CDF from Equation (4.6) and the empirical risk for the PDF from Equation (4.8) we obtain the following empirical risk:

$$R_{L^2,\overline{L}^2}^{\mathcal{S}_{\mathcal{X}}}(F) = \frac{1}{I}\sum_{i=0}^{I-1}\left(F^*_{\text{emp}}(\boldsymbol{x}_i) - F(\boldsymbol{x}_i)\right)^2 - \frac{2}{I}\sum_{i=0}^{I-1} f(\boldsymbol{x}_i) + Q(f^2), \tag{4.9}$$

where we recall that the density function f can be written in terms of the distribution function F as:

$$f(\boldsymbol{x}) = \left.\frac{\partial^n F}{\partial x^{(1)}\cdots\partial x^{(n)}}\right|_{\boldsymbol{x}}. \tag{4.10}$$

In this way, we have defined a distance which does not make use of the labels. For a more in depth discussion of the election of the two risk functions see (Manzano, 2024).

[1] The Glivenko–Cantelli theorem states that the empirical CDF converges uniformly to the true CDF almost surely (Cantelli, 1933).

The solution of this minimization problem gives us both the distribution and density functions, which allows us to compute different risk metrics. One of the most popular ones being the VaR. This is an estimate of how much one can lose from one's portfolio over a given time horizon, with a given degree of confidence (Wilmott, 2007). More specifically, given a confidence level $\alpha \in (0,1)$, the VaR of a portfolio can be defined as,

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

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

In the next section, we will mainly focus in the first part of recovering the full shape of the distribution and density functions, since it is a broader problem and the most of the computational resources are expended here and we will make a brief comment on the computation of the VaR.

4.3. Numerical experiments in finance

In this section we will compare the performance of the two empirical risks $R_{L^2}^{\mathcal{S}_\mathcal{X}}$ and $R_{L^2,\overline{L}^2}^{\mathcal{S}_\mathcal{X}}$ in the context of finance. For this purpose we choose two of the most popular distribution functions in finance: the normal and lognormal distributions. The CDF and PDF of the normal distribution with mean μ and and variance σ^2 are, respectively

$$F_{\text{N}}^*(x) = \frac{1}{2}\left[1 + \text{erf}\left(\frac{x-\mu}{\sigma\sqrt{2}}\right)\right], \quad f_{\text{N}}^*(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}. \tag{4.11}$$

The CDF and PDF of the lognormal distribution with parameters $\overline{\mu}$ and $\overline{\sigma}$ are:

$$F_{\text{LN}}^*(x) = \frac{1}{2}\left[1 + \text{erf}\left(\frac{\ln(x)-\overline{\mu}}{\overline{\sigma}\sqrt{2}}\right)\right], \quad f_{\text{LN}}^*(x) = \frac{1}{x\sigma\sqrt{2\pi}} \exp\left(-\left(\frac{\ln(x)-\overline{\mu}}{\overline{\sigma}\sqrt{2}}\right)^2\right). \tag{4.12}$$

In finance, the lognormal distribution has a special prevalence through the Black-Scholes model, which assumes that the evolution of prices follow a Geometric Brownian Motion (GBM), that satisfies the SDE:

$$dX_t = \mu_{\text{BS}} X_t dt + \sigma_{\text{BS}} X_t dW_t. \tag{4.13}$$

The distribution of prices for a given time T are lognormal with parameters:

$$\overline{\mu} = (\mu_{\text{BS}} - 0.5\sigma_{\text{BS}}^2)T, \quad \overline{\sigma} = \sigma_{\text{BS}}\sqrt{T}\,. \tag{4.14}$$

The first experiment that we conducted to compare the performance under the different loss functions is shown in

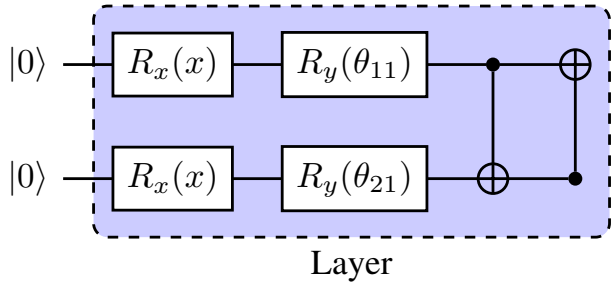

***Figure 15**: One layer of the PQC used in the experiments. For the experiments we use a total of three layers. The parameters θ_{ij} are variational parameters.*

Figure 16. All simulations have been performed using 10 samples of the normal distribution with mean zero and variance one. Each experiment has been repeated 100 times and we depict the 25, 50 and 75 percentiles in colored solid lines. As we can see in Figure 16 the results obtained by using the differential machine learning approach are much better than the ones obtain by just mapping to the empirical CDF. On the one hand, because the variance is smaller when using the L_h loss function. On the other hand, because we obtain a much better performance on the tails.

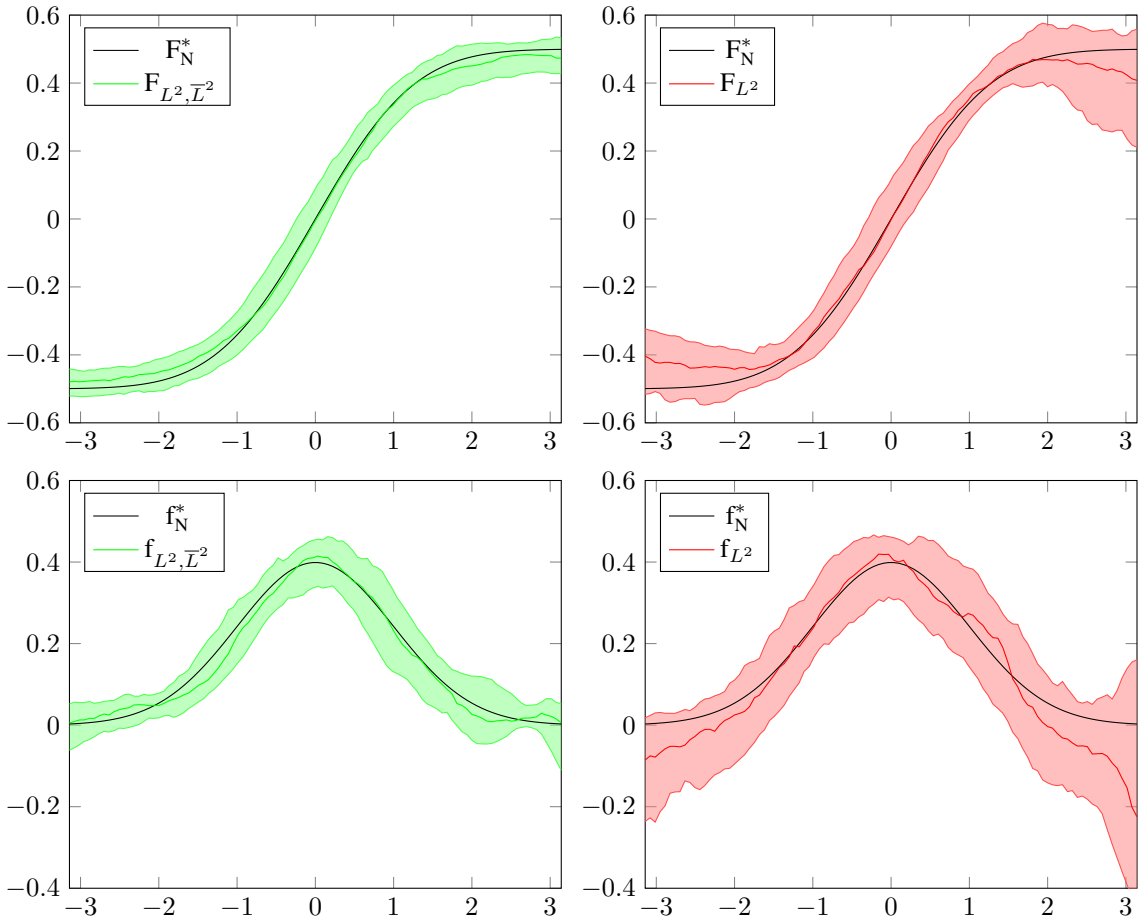

***Figure 16**: In this picture we have trained the PQC of Figure 15 with three layers to approximate the standard normal distribution (i.e., with $\mu = 0$ and $\sigma = 1$), using the two different empirical risk functions $R_{L^2}^{\mathcal{S}_\mathcal{X}}$ and $R_{L^2,\overline{L}^2}^{\mathcal{S}_\mathcal{X}}$. We have used 10 training points (10 samples of the distribution) and 100 epochs with the Adam optimizer. The experiments have been repeated 100 times. We have normalised the data to lie in the interval $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$.*

In Table 2 we also show the result of the computation of the 95-VaR for the returns of the standard normal distribution. As we can see we obtain much better results when we use the $R_{L^2,\overline{L}^2}^{\mathcal{S}_\mathcal{X}}$ loss function, as we expeted by the fitting of the distribution from Figure 16.

Loss	Exact	Mean	Confidence Interval	25-50-75 Percentiles
$R_{L^2,\overline{L}^2}^{\mathcal{S}_\mathcal{X}}$	2.64	2.68	(2.46, 2.90)	$3.14 - 2.71 - 2.33$
$R_{L^2}^{\mathcal{S}_\mathcal{X}}$	2.64	2.07	$(1, 15, 2.99)$	$3.14 - 3.05 - 2.21$

***Table 2**: Results for the 95-VaR when we use the PQC of Figure 16. The mean VaR is computed with a 95% confidence interval using Chebysev bounds. The percentiles are the $25 - 50 - 75$.*

The second experiment that we conduct to compare the performance under the different loss functions is shown in Figure 17. All simulations have been performed using 250 samples of the lognormal distribution with $\mu_{\text{BS}} = 0.05$ and $\sigma_{\text{BS}} = 0.5$ and maturity 0.5. Each experiment has been repeated 100 times and we depict the 25, 50 and 75 percentiles in colored solid lines.

The results from Figure 17 follow the same trend as the ones from Figure 16. The biggest impact can be seen in the tails, where we obtain a much better performance by using the differential loss function.

In Table 3 we show the result of the computation of the 95-VaR for the returns of the lognormal distribution. In this case, we did not observe such a big impact in the loss function, although there is some improvement.

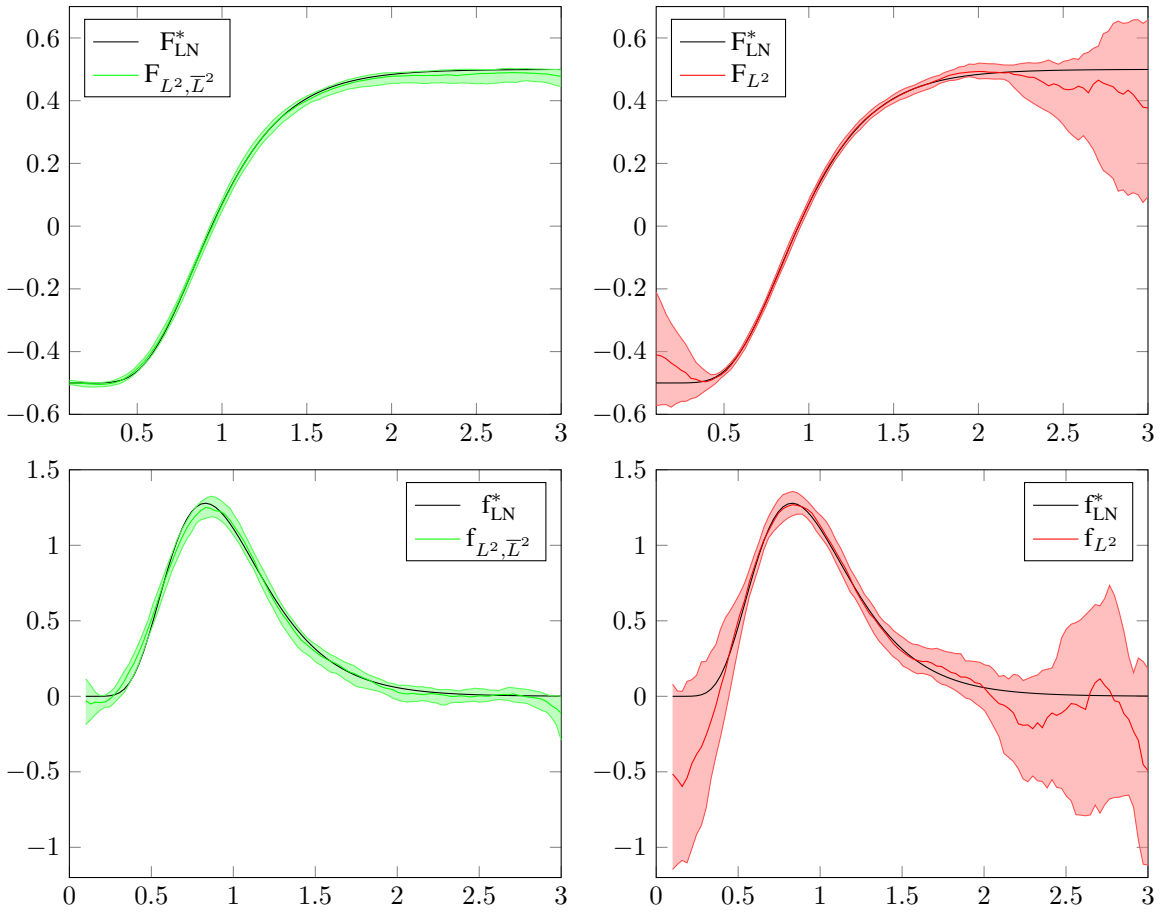

***Figure 17**: In this picture we have trained the PQC of Figure 15 to approximate the lognormal distribution with parameters $\mu_{BS} = 0.05$, $\sigma_{BS} = 0.5$ and $T = 0.5$ using the two different empirical risk functions $R^{\mathcal{S}_\mathcal{X}}_{L^2}$ and $R^{\mathcal{S}_\mathcal{X}}_{L^2,\overline{L}^2}$. We have used 250 training points (250 samples of the distribution) and 150 epochs with the Adam optimizer. The experiments have been repeated 100 times. We have normalised the data to lie in the interval $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$.*

Loss	Exact	Mean	Confidence Interval	25-50-75 Percentiles
$R^{\mathcal{S}_\mathcal{X}}_{L^2,\overline{L}^2}$	0.47	0.70	(0.56, 0.84)	$0.99 - 0.77 - 0.42$
$R^{\mathcal{S}_\mathcal{X}}_{L^2}$	0.47	0.76	$(0.64, 0, 88)$	$0.99 - 0.99 - 0.49$

***Table 3**: Results for the 95-VaR when we use the PQC of Figure 17. The mean VaR is computed with a 95% confidence interval using Chebysev bounds. The percentiles are the $25 - 50 - 75$.*

5.Conclusions

The exploration of quantum computing techniques for quantitative finance within the NEASQC project has yielded significant insights and advancements. However, the technology has not yet reached a level of maturity where it can be effectively integrated into industrial applications. The current hardware constraints and the state of quantum computing present substantial challenges.

One of the primary obstacles encountered is the significant technological gap, which prevents a direct comparison between classical and quantum algorithms. This gap highlights the need for further development and refinement of quantum hardware and algorithms. Despite these challenges, the project has made notable progress in several areas. We have developed new techniques for Quantum Accelerated Monte Carlo (QAMC) and Quantum Machine Learning (QML). The advancements in QAMC are particularly relevant from a practitioner's perspective, as the provided pipeline closely resembles the one used by financial institutions such as our collaborator HSBC. Specifically, the fact that we do not need to separate our Monte Carlo problem into co-domains greatly simplifies the future integration of our libraries with existing financial libraries. Additionally, we have made significant progress in understanding new state-of-the-art techniques such as PQCs and DML from the perspective of practical applications.

This part of the project has also produced five research papers (Gómez et al., 2022; Manzano et al., 2022; Manzano, Dechant, et al., 2023; Manzano, Ferro, et al., 2023; Manzano, Musso, & Leitao, 2023) contributing valuable knowledge to the field of quantum computing in finance. Additionally, a comprehensive software library QQuantLib (Ferro et al., 2024) has been developed, providing implementations of the new techniques and facilitating further research and experimentation.

Some of the techniques developed need to be tested in practice as our computational resources grow. It is challenging to make a theoretical case for their effectiveness without practical implementation and validation.

In conclusion, while significant progress has been made in developing quantum algorithms for pricing and VaR estimation, these algorithms are not yet competitive with classical algorithms when executed on current Noisy Intermediate-Scale Quantum (NISQ) architectures. The primary challenges moving forward are twofold. First, the execution of deep circuits required by amplitude estimation routines is not feasible without error correction techniques. Second, the direct translation of classical circuits for Stochastic Differential Equation (SDE) simulations into quantum circuits requires a number of qubits beyond current NISQ capabilities (see Appendix A).

List of Acronyms

Term	Definition
AdaptQAE	Adaptive Quantum Amplitude Estimation
AE	Amplitude Estimation
BAA	Bounded Approximation Algorithm
CLF	Conventional Likelihood Function
CMC	Classical Monte Carlo
CQPEAE	Amplitude Estimation based on Classical Quantum Phase Estimation
CVA	Credit Valuation Adjustment
DD	Decision Diagram
EIS	Exponential Increasing Schedule
ELF	Engineered Likelihood Function
FAE	Faster Amplitude Estimation
FT3	FinisTerrae III
GBM	Geometric Brownian Motion
IQPEAE	Amplitude Estimation based on Iterative Quantum Phase Estimation
IQAE	Iterative Quantum Amplitude Estimation
LIS	Linear Increasing Schedule
LRSP	Low-rank state preparation
MC	Monte Carlo
MLAE	Maximum Likelihood Amplitude Estimation
MPS	Matrix Product States
NEASQC	NExt ApplicationS of Quantum Computing
NISQ	Noisy Intermediate-Scale Quantum
PDE	Partial Differential Equation
PDF	Probability Distribution Function
PE	Phase Estimation
PQC	Parametric Quantum Circuit
QAMC	Quantum Accelerated Monte Carlo
QFT	Quantum Fourier Transformation
qGAN	Quantum Generative Adversarial Network
QLM	Quantum Learning Machine
QPU	Quantum Processing Unit
QQuantLib	Quantum Quantitative Finance Library
QSP	Quantum Signal Processing
QSVT	Quantum Singular Value Transformation
RandomQA	Random Quantum Amplitude Estimation
RC	Randomised Compiling
RMSE	Root Mean Square Error
RQAE	Real Quantum Amplitude Estimation
SDE	Stochastic Differential Equation
SSQW	Split Step Quantum Walk
SVD	Singular Value Decomposition
UC	Use Case
VaR	Value at Risk

***Table 4**: Acronyms and Abbreviations*

List of Figures

List of Tables

Bibliography

Abrams, D. S., & Williams, C. P. (1999). Fast quantum algorithms for numerical integrals and stochastic processes.

Atos. (2016). Myqlm. https://myqlm.github.io/

Bergholm, V., Izaac, J., Schuld, M., Gogolin, C., Blank, C., McKiernan, K., & Killoran, N. (2018). Pennylane. https://pennylane.ai/

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

Brassard, G., Høyer, P., Mosca, M., & Tapp, A. (2002). Quantum amplitude amplification and estimation. https://doi.org/10.1090/conm/305/05215

Callison, A., & Browne, D. E. (2022). Improved maximum-likelihood quantum amplitude estimation. https://doi.org/10.48550/ARXIV.2209.03321

Cantelli, F. P. (1933). Sulla determinazione empirica delle leggi di probabilità. *Giorn. Ist. Ital. Attuari*, *4*, 421–424.

Egger, D. J., Gutiérrez, R. G., Mestre, J. C., & Woerner, S. (2020). Credit risk analysis using quantum computers. *IEEE Transactions on Computers*.

Ferro, G., Manzano, A., & Musso, D. (2024). Quantum quantitative finance library (qquantlib). https://github.com/NEASQC/FinancialApplications

Fukuzawa, S., Ho, C., Irani, S., & Zion, J. (2023, January). Modified iterative quantum amplitude estimation is asymptotically optimal. In *2023 proceedings of the symposium on algorithm engineering and experiments (ALENEX)* (pp. 135–147). Society for Industrial; Applied Mathematics.

Giurgica-Tiron, T., Kerenidis, I., Labib, F., Prakash, A., & Zeng, W. (2022). Low depth algorithms for quantum amplitude estimation. *Quantum*, *6*, 745. https://doi.org/10.22331/q-2022-06-27-745

Gómez, A., Leitao Rodriguez, A., Manzano, A., Nogueiras, M., Ordóñez, G., & Vázquez, C. (2022). A survey on quantum computational finance for derivatives pricing and VaR. *Archives of Computational Methods in Engineering*, *29*, 4137–4163. https://doi.org/10.1007/s11831-022-09732-9

Grinko, D., Gacon, J., Zoufal, C., & Woerner, S. (2021). Iterative quantum amplitude estimation. *npj Quantum Information*, *7*(1). https://doi.org/10.1038/s41534-021-00379-1

Huge, B., & Savine, A. (2020). Differential machine learning. *arXiv*. https://doi.org/10.48550/arXiv.2005.02347

Hull, J. (1997). *Options, futures, and other derivatives* (3. ed., internat. ed). Prentice Hall. http://gso.gbv.de/DB=2.1/CMD?ACT=SRCHA&SRT=YOP&IKT=1016&TRM=ppn+216058376&sourceid=fbw_bibsonomy

Manzano, A., Musso, D., Leitao, Á., Gómez, A., Vázquez, C., Ordóñez, G., & Nogueiras, M. (2022). A modular framework for generic quantum algorithms. *Mathematics*, *10*, 785.

Manzano, A. (2024). *Contributions to the pricing of financial derivatives contracts in commoditiy markets and the use of quantum computing in finance* [Doctoral dissertation, Universidade da Coruña]. http://hdl.handle.net/2183/39574

Manzano, A., Dechant, D., Tura, J., & Dunjko, V. (2023). Parametrized quantum circuits and their approximation capacities in the context of quantum machine learning. https://arxiv.org/abs/2307.14792

Manzano, A., Ferro, G., Leitao, Á., Vázquez, C., & Gómez, A. (2023). Real option pricing using quantum computers. *arXiv. Submitted for publication*.

Manzano, A., Musso, D., & Leitao, Á. (2023). Real quantum amplitude estimation. *EPJ Quantum Technology*, *10*(1), 1–24.

Montanaro, A. (2015). Quantum speedup of Monte Carlo methods. https://doi.org/http://doi.org/10.1098/rspa.2015.0301

Nakaji, K. (2020). Faster amplitude estimation. *Quantum Information and Computation*, *20*(13&14), 1109–1123. https://doi.org/10.26421/qic20.13-14-2

Nielsen, M. A., & Chuang, I. L. (2011). *Quantum computation and quantum information: 10th anniversary edition*. Cambridge University Press.

Pérez-Salinas, A., Cervera-Lierta, A., Gil-Fuster, E., & Latorre, J. I. (2020). Data re-uploading for a universal quantum classifier. *Quantum*, *4*, 226.

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

Schuld, M., Sweke, R., & Meyer, J. J. (2021). Effect of data encoding on the expressive power of variational quantum-machine-learning models. *Physical Review A*, *103*(3), 032430. https://doi.org/10.1103/PhysRevA.103.032430

Stamatopoulos, N., Egger, D. J., Sun, Y., Zoufal, C., Iten, R., Shen, N., & Woerner, S. (2020). Option pricing using quantum computers. *Quantum*, *4*, 291.

Suzuki, Y., Uno, S., Raymond, R., Tanaka, T., Onodera, T., & Yamamoto, N. (2020). Amplitude estimation without phase estimation. *Quantum Information Processing*, *19*(2). https://doi.org/10.1007/s11128-019-2565-2

Wilmott, P. (2007). *Paul wilmott introduces quantitative finance* (2nd ed.). Wiley-Interscience.

Zhao, Y., Wang, H., Xu, K., Wang, Y., Zhu, J., & Wang, F. (2022). Adaptive algorithm for quantum amplitude estimation. https://doi.org/10.48550/ARXIV.2206.08449

A.Description of the QAMC experiments

In Section 3.1, we have described the general setup of QAMC for pricing. A rough estimation indicates that we would require the order of hundreds or thousands of qubits to build the algorithm following the same steps as its classical counterpart. With the current hardware, this is not feasible. Hence, in order to conceptually test this technique, we need to perform several simplifications.

If we assume only European payoffs we can make the first simplification since we do not need to store the whole paths for the underlying. Instead, we will consider that we have just one register which encodes the value of the underlying.

The next simplification would be reducing as much as the depth of the quantum circuit which loads the probability distribution. In practice this means that we need to restrict ourselves to models where we know the analytical form of the underlying distribution. Thus, in this work, we will consider the classical (and well-known) model given by the following Black-Scholes SDE under the risk neutral measure (Black & Scholes, 1973)

$$dS_t \;=\; rS_t\,dt \;+\; \sigma S_t\,dW_t, \tag{A.1}$$

where r denotes the risk-free rate, σ is the volatility of the underlying asset price and W_t denotes a Brownian motion in a particular probability space, so that W_t follows a $\mathcal{N}(0,t)$ distribution (and its increment dW_t follows a $\mathcal{N}(0,dt)$).. As the expression of the probability distribution generated by the SDE (A.1) is known, we assume that we have a unitary U_{BS} which encodes the Black-Scholes distribution p_{BS} in some predefined points. Note that, regardless all the simplifications, the algorithm is conceptually the same: we have a quantum circuit specified by the oracle $U_S = U_{BS}$ which generates samples for the underlying price at maturity, S_T, with the correct probability distribution.

For all the experiments in Section 3 we have encoded the Black-Scholes probability distribution with risk-free rate 0.01 and volatility 0.5. Moreover, we have considered a one year maturity and an initial underlying value of 1.0. For the discretization of the distribution we have considered 32 points between 0.01 and 5.0 which requires the use of 5 qubits. For any other content, we have kept the general setting.