

Quantum Internal Models for Solvency II and Quantitative Risk Management

1. Abstract

In this paper, I extend the methodologies proposed by Egger et al. to a substantial, real-world challenge: constructing a Quantum Internal Model for a medium-sized insurance company. Leveraging the author's extensive experience as the former Head of Internal Model at a prominent UK insurer, I closely examine the practical bottlenecks in developing and maintaining Quantum Internal Models. My work seeks to determine whether the quadratic speedup, through Quantum Amplitude Estimation (Egger et al., 2019)[1], can be realized for problems at an industrial scale. Additionally, I identify both the obstacles and the potential opportunities that emerge from applying quantum computing to the field of insurance risk management.

2. Introduction

Over the past decade, significant advancements in quantum computing have culminated in the deployment of prototype quantum computers accessible via cloud platforms to a global audience. This accessibility has empowered a broad spectrum of individuals, ranging from Quantum Information Scientists to Quantum Software Engineers and enthusiasts, to gain practical experience with quantum processes. Currently, we find ourselves in the Noisy Intermediate-Scale Quantum (NISQ) era, a phase characterized by quantum computers that are theoretically capable of demonstrating quantum advantage but are hindered by high noise levels, making the achievement of quantum supremacy in many applications unattainable (Preskill, 2018)[1].

Recent research by Egger et al. (2019)[2] in "Quantum Risk Management" and their subsequent 2020 study in IEEE Transactions on Quantum Engineering, "Quantum Computing for Finance,"[3] illustrate that quantum computers can achieve a quadratic speedup over classical counterparts for specific, small-scale models. These investigations, however, were limited to stylized problems and did not extend to real-world applications.

This paper provides a high-level introduction to Internal Models (For UK Solvency II firms) and quantum computing, as necessary background for building a quantum internal model. Then, this paper extends previous research [2,3], by assessing whether a quantum advantage exists for practical applications such as a Value at Risk

calculation for a sufficiently large and complex portfolio. In particular, I answer the following questions:

- 1) How is a Quantum Internal model different from the types of problems discussed in previous research?
- 2) How the methods used for solving stylised problems in previous papers can be extended for building a Quantum Internal Model?
- 3) Whether a Quantum Advantage exists for such problems?
- 4) And, if there is scope for a quantum advantage in other, similar problems?

3. Internal Models for Solvency II firms

3.1 Solvency II

Solvency II is a regulatory framework for insurance firms within the European Union. It came into effect on 1 January 2016. The directive aims to ensure that insurance companies are financially sound and capable of meeting their obligations to policyholders. Solvency II consists of 3 main pillars, Quantitative Requirements (Pillar 1), Governance and Supervision (Pillar 2) and Disclosure and Transparency (Pillar 3).

Internal Models fit within Pillar 1 of the Solvency II framework, which deals with the quantitative requirements that insurance and reinsurance companies must meet. Under Solvency II, firms are required to hold enough capital to ensure that they can withstand financial shocks and meet their obligations to policyholders over a one-year period with a high degree of certainty. This required level of capital is known as the Solvency Capital Requirement (SCR).

Pillar 1 offers two main approaches for calculating the SCR: the Standard Formula and the use of Internal Models.

1. **Standard Formula:** This is a one-size-fits-all approach provided by the regulatory framework that applies broadly to all insurance firms. It uses a set methodology to calculate the SCR based on predefined risk categories and parameters.
2. **Internal Models:** These are bespoke, sophisticated models developed by insurance firms themselves to calculate their SCR. The use of an Internal Model is subject to the approval of the relevant supervisory authority. The model must meet specific criteria and demonstrate that it is capable of accurately reflecting the firm's risk profile.

Internal Models offer several potential advantages over the Standard Formula:

- They can provide a more accurate and sensitive measure of the risks faced by a specific firm, taking into account its unique characteristics, risk profile, and business model.
- They can encourage better risk management practices, as firms that develop and use Internal Models typically have a deeper understanding of their risk exposures and how they can be managed.
- They can potentially lead to a lower SCR if the model accurately captures the risk profile and demonstrates lower risk than the conservative assumptions under the Standard Formula.

However, the development, validation, and maintenance of Internal Models require significant resources and expertise. Moreover, regulatory approval for using an Internal Model is rigorous, requiring extensive documentation, testing, and evidence that the model meets the Solvency II requirements for accuracy, appropriateness, and comprehensiveness in capturing the firm's risk profile.

3.2 Internal Models

The primary function of an Internal Model is to compute the Solvency Capital Requirement (SCR), which employs a Value at Risk (VaR) methodology to quantify the capital needed to cover potential losses over a one-year horizon with a 99.5% confidence level. An Internal Model typically encompasses three main components:

1. **Individual Risk Modules:** These modules typically utilize parametric distributions to model the behaviour of singular risk factors. For instance, the Lognormal distribution might be employed to simulate equity price returns. This approach allows for a nuanced analysis of each risk factor in isolation, facilitating a more granular understanding of potential vulnerabilities.
2. **Dependency Module:** This component typically employs a copula approach to assess the correlation and tail dependence among different risk factors. It determines the extent to which shocks in one area, such as equity prices or credit spreads, are likely to coincide with extreme events in another, along with the individual risk modules, the dependency module determines the joint distribution function of the risks.
3. **Asset and Liability Valuation Module:** This module converts the scenarios generated from the joint distribution of risks into impacts on Own Funds or the Net Asset Value, facilitating the calculation of the 99.5th percentile VaR. It often relies on polynomial based proxy models that approximate the effects on the present value of multi-year cash flows across a vast array of scenarios efficiently.

The use of proxy models enables the Internal Model to assess balance sheet impacts under numerous scenarios within practical timeframes.

Together, the individual risk and dependency modules form the core of the Monte Carlo engine. This engine performs several hundred thousand to over a million simulations, projecting potential outcomes over the forthcoming year. This extensive simulation process provides a comprehensive view of the potential volatility and risk exposure, enabling insurers to estimate the necessary capital requirements accurately.

The integration of these modules within an Internal Model offers a robust framework for understanding and managing the complex interplay of risks faced by insurance companies, ensuring they maintain sufficient capital to withstand adverse events while optimising their financial and operational strategies.

The Asset and Liability valuation module then translates these scenarios into Net Asset Value impacts (or Profit and Loss PnL), that can then be used to calculate the 99.5th percentile VaR. These asset and liability valuation models (typically polynomials based proxy models) are then used to evaluate the balance sheet in each of the simulated scenarios.

Polynomial proxy models are frequently employed due to their capacity to significantly accelerate balance sheet evaluations compared to the more computationally intensive cash flow models, while maintaining simplicity. These models can be adeptly calibrated to accurately mirror the valuation of any asset or liability, or even the Net Asset Value (NAV) of portfolios, directly. The calibration of polynomial proxy models is a sophisticated, multi-stage endeavour, involving the adjustment of both the structural components (such as order and cross-terms) and the coefficients of the polynomial terms. While a detailed discussion of the calibration process is beyond the scope of this paper, it is pertinent to acknowledge that the foundational motivation for employing polynomials in proxy modelling is rooted in the Weierstrass approximation theorem [4]. This theorem asserts that any continuous function defined on a closed interval can be uniformly approximated as closely as desired by a polynomial function. This principle underpins the theoretical justification for the use of polynomial models in financial mathematics and computational finance.

The above structure lends itself nicely also to a Quantum implementation, which is discussed in the next section.

Egger et al [3][3], have discussed the application of Quantum Computing to Value at Risk calculations, and have shown that a 'quadratic speedup' is possible with a quantum algorithm known as the 'Quantum Amplitude Estimation' algorithm for calculating VaR.

4. Quantum Computing

Quantum computing represents a paradigm shift in computational technology, diverging fundamentally from the classical computing frameworks that form the backbone of our digital world. Although the physical realizations of quantum computers differ among providers, it is essential to explore the distinctions between quantum and classical computing from a software perspective, focusing on the unique methods of information encoding and manipulation.

Classical computing relies on binary bits as the basic unit of data, which can exist in one of two states: 0 or 1. This binary system underpins all classical computation, enabling the processing and storage of vast amounts of data. In contrast, quantum computing introduces the concept of the quantum bit, or qubit, which leverages principles of quantum mechanics to exist in multiple states simultaneously through a phenomenon known as superposition. This ability allows a qubit to represent a 0, a 1, or any combination (quantum superposition) of these states, offering a new dimension of computational depth. We can visualise this by imagining with classical computing, elementary unit of information (bit) can only be the north or the south pole of the Earth, whereas with quantum computing, the elementary units (qubits) can represent a position anywhere on the surface of a sphere.

Apart from the 'information capacity' of quantum bits (qubits), quantum computers can exploit quantum phenomena such as interference and entanglement. When qubits become entangled, the state of one qubit instantly influences the state of another, no matter the distance separating them. Similarly, quantum interference allows computations to amplify correct paths to a solution while cancelling out incorrect ones. These properties enable quantum systems to evolve their states at unprecedented speeds, by exploiting parallelism that is not possible in classical systems.

However, the current era of quantum computing, known as the Noisy Intermediate-Scale Quantum (NISQ) era, is characterized by quantum computers that are powerful yet imperfect. These machines are prone to errors due to quantum noise and decoherence, limiting their immediate practical applications. Despite these challenges, ongoing advancements in quantum error correction and algorithm development continue to push the boundaries of what is computationally feasible.

In summary, the key differences between quantum and classical computing stem from the fundamental principles of quantum mechanics that quantum computers exploit: superposition, entanglement, and interference. These principles enable quantum computers to process information in ways that classical computers cannot, offering the potential for significant breakthroughs in various fields, including cryptography, materials science, and complex system simulation. As the technology matures and

becomes more accessible, it is anticipated that quantum computing will complement classical computing, tackling problems that are currently beyond reach.

4.1 Gate based model of quantum computation.

The gate-based model of quantum computation is the most widely recognized and developed model for quantum computing. It is analogous to the classical model of computation using logic gates, but with quantum gates operating on quantum bits (qubits) instead of classical bits. Here's a more detailed description:

Gate-Based Quantum Computing:

- **Qubits:** The basic unit of information is the qubit, which can be in a state of 0, 1, or any quantum superposition of these states. Multiple qubits can also be entangled, a phenomenon where the state of one qubit can depend on the state of another, no matter the distance between them.
- **Quantum Gates:** These are the building blocks of quantum algorithms, analogous to classical logic gates. However, quantum gates manipulate qubit states using operations like superposition and entanglement. The operations are reversible, and each gate represents a unitary transformation.
- **Quantum Circuits:** Quantum algorithms are constructed as quantum circuits, which are sequences of gate operations that evolve the state of qubits to perform computations.
- **Measurement:** After the gates are applied, the qubits are measured, collapsing their superposition states into definite classical states (0 or 1), providing the output of the computation.

This model is powerful because it can theoretically execute any quantum (or classical) algorithm, making it a universal model for computation.

If a gate-based quantum computer is the quantum analogue for a digital computer, then quantum annealers are analogous to analogue classical computers (such as a Differential Analyzer or a Slide Rule).

Quantum Annealing:

- **Physical Process:** Quantum annealing is inspired by the physical process of annealing in metallurgy where materials are heated and then slowly cooled to remove defects. It uses quantum fluctuations instead of thermal fluctuations.
- **Specialized Problem-Solving:** It's specifically designed to solve optimization problems by finding the lowest energy state of a system, which corresponds to the optimal solution of the problem.

- **Adiabatic Quantum Computing:** Adiabatic computing is a subset of quantum computing, which is based on the principle that a quantum system remains in its ground state if the Hamiltonian that describes it changes slowly enough.
- **Not Universal:** While powerful for optimization, quantum annealing isn't known to be universal for quantum computation and is tailored to specific types of problems rather than general purpose computation.

Contrast Between Gate-Based and Quantum Annealing:

- **Universality:** Gate-based quantum computing is a universal model capable of performing any computational task that quantum mechanics allows, while quantum annealing is specialized for optimization problems.
- **Algorithms:** The types of algorithms that each can run differ significantly. Gate-based quantum computing can implement any algorithm designed for quantum computers, while quantum annealing is limited to those that can be mapped to an optimization problem.
- **Hardware Implementation:** Quantum annealing is typically implemented in quantum annealers like D-Wave systems, which are designed to solve optimization problems. In contrast, gate-based quantum computers require a set of universal quantum gates and typically aim to be more general-purpose.
- **Error Correction:** Gate-based models have developed error correction protocols to deal with the fragility of quantum states, which is crucial for their scalability. Quantum annealing also deals with noise and errors but in a different manner, often leveraging the robustness of the ground state search process itself.

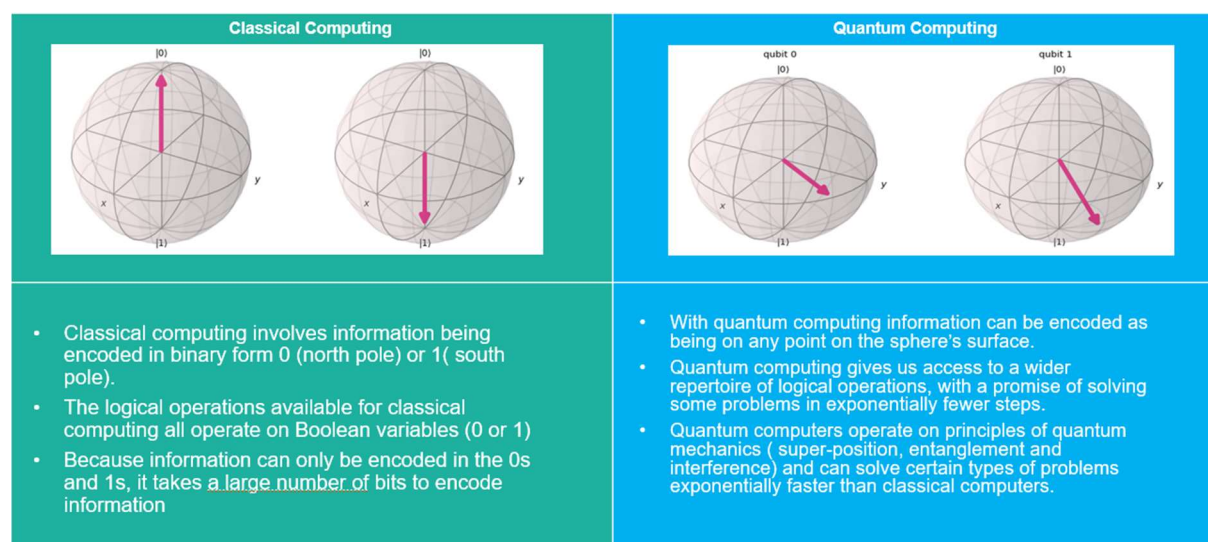
In summary, the gate-based model is a general-purpose approach to quantum computing, analogous to modern digital computers, and capable of running a wide range of algorithms, including those not yet imagined. Quantum annealing, while potentially powerful for certain applications, is more specialized and does not provide a universal computational model.

4.2 Quantum vs Classical bits

Classical computers operate using bits as the basic unit of information, which can exist in one of two states: 0 or 1. The processing of information is deterministic, meaning the output of computation is determined exactly, given the input and the operation performed. Classical computing relies on logical operations, or gates (such as AND, OR, and NOT), which manipulate bits to perform calculations and process data.

Quantum computers, on the other hand, use quantum bits or qubits as their fundamental unit of information. Unlike classical bits, qubits can exist in a state of **superposition**, allowing them to represent both 0 and 1 simultaneously. This capability, along with **entanglement**—a phenomenon where the state of one qubit can depend on the state of another, no matter the distance between them—enables quantum computers to perform a vast number of calculations at once. Quantum computation is probabilistic rather than deterministic, with the outcome of calculations being a probability distribution over all possible states. This means that given the input and the operations performed, the output is a probability distribution, rather than an exact value.

Figure 1: Difference between classical bits and quantum bits at a glance.



4.3 Quantum vs Classical Gates

Classical Gates: In classical computing, gates are simple and deterministic logic circuits that process two states, 0 and 1. These gates are the building blocks of more complex circuits that execute algorithms and perform computational tasks. Each gate type performs a specific logical operation, combining them in various sequences can in theory solve any computable problem.

Quantum Gates: Quantum gates manipulate qubits through operations that reflect the principles of quantum mechanics. Unlike their classical counterparts, quantum gates

can exploit superposition and entanglement, enabling a single operation to perform complex transformations on an exponential number of states simultaneously. Quantum gates are represented mathematically by unitary matrices, ensuring that all operations are reversible—a stark contrast to some classical operations like the AND gate, which cannot uniquely determine inputs from the output. This reversibility is crucial for preserving information in quantum algorithms.

Quantum Gates (Examples)	Description
Hadamard (H)	Creates a superposition of
Pauli-X (Quantum NOT)	Flips the state of a qubit, analogous to a classical NOT but with quantum properties.
Pauli-Y	Performs a rotation around the Y-axis on the Bloch sphere, also applying a complex phase.
Pauli-Z	Introduces a phase flip, leaving
CNOT	Conditional NOT, flips the target qubit only if the control qubit is in the
Toffoli (CCNOT)	Acts as a controlled-controlled NOT, similar to a classical AND gate with a quantum twist.
SWAP	Swaps the quantum states of two qubits.
S (Phase Gate)	Adds a phase of $\pi/2$ to the
T ($\pi/8$ Gate)	Adds a phase of $\pi/4$ to the
Phase (ϕ Gate)	Applies a phase shift of ϕ to the
SQRT(NOT)	A unique quantum operation that represents the square root of a NOT gate, something without a classical analogue.
Controlled (U)	Applies a unitary operation U to the target qubit, contingent on the state of the control qubit.

Quantum gates can perform all operations that classical gates do and more, but they're not directly analogous to classical gates. Instead, they are representations of unitary operations that can be applied to qubits. They enable complex transformations that are crucial for quantum algorithms. Classical gates, on the other hand, are typically irreversible (except for the NOT gate) and manipulate bits deterministically based on Boolean logic.

Classical Gates (Examples)	Description
NOT	Inverts the bit value: 0 becomes 1, and 1 becomes 0.
AND	Outputs 1 if both inputs are 1, otherwise 0.

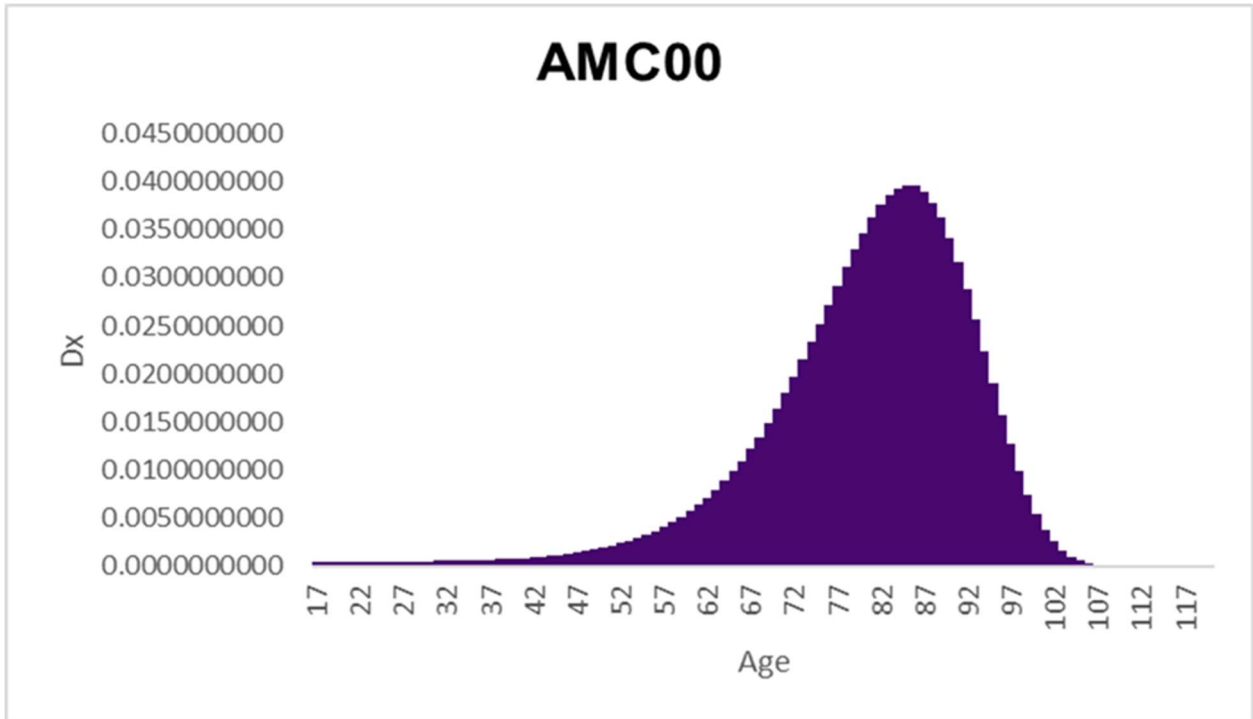
OR	Outputs 1 if at least one input is 1.
XOR	Outputs 1 if the inputs are different, otherwise 0.
NAND	Outputs 0 only if both inputs are 1, otherwise 1 (inverse of AND).
NOR	Outputs 1 only if all inputs are 0, otherwise 0 (inverse of OR).
XNOR	Outputs 1 if the inputs are the same, otherwise 0 (inverse of XOR).
Multiplexer	Directs one of several input signals to a single output line based on control signals.
Demultiplexer	Takes a single input and channels it to one of several output lines based on control signals.
Encoder	Converts information from one format or code to another, typically from simple to more complex formats.
Decoder	Reverses the encoding process, transforming complex coded signals back to their original form.
Shift	Moves all bits in a binary number to the left or the right, filling in the new space with zeros.

Classical gates deal with definite states, adhering strictly to binary logic. In contrast, Quantum gates, such as the Hadamard gate, create superpositions of qubits, while others, like the CNOT gate, entangle qubits, allowing quantum algorithms to solve certain problems more efficiently than classical algorithms can. The difference in gate functionality underpins the quantum advantage in specific computational tasks, offering exponential speedups in some cases.

4.4 An actuarial example

Given the main application being explored in this paper is the calculation of the SCR for an insurance firm, an insurance inspired example is useful. The figure below shows the probabilities of ‘death’ at different ages for a new-born. Such tables are typically used for various actuarial calculations, such as calculating the price of a life insurance policy, or an annuity.

Figure 2: An actuarial ‘life’ table.

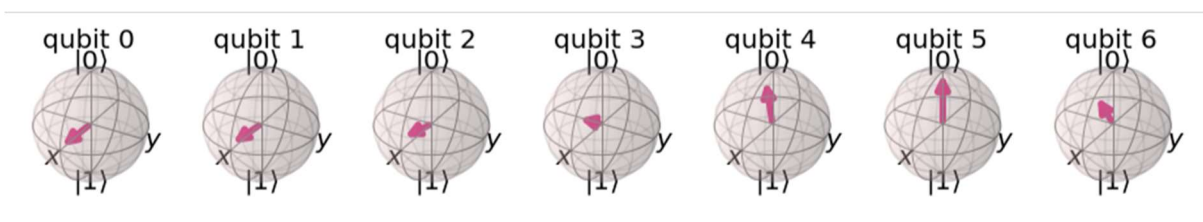


To illustrate the difference between the power of quantum computers vs classical computers, let's consider how many qubits(bits) are required to encode the information in the above table.

There are c120 ages plotted on the chart, this would require $120 \times 64 = 7680$ bits, if encoded the probabilities in double-precision.

By contrast, the above probability distribution can be captured using just 7 logical qubits ($2^7=128$), shown below.

Figure 3: Actuarial 'life' table encoded in 7 qubits.



Interestingly, sampling from above distribution using a classical computer requires further bits for implementing the Monte-Carlo sampling. In the case of quantum

computation, we can just create the state encoding the above distribution, and execute the circuit several times, i.e. we get the randomness for ‘free’.

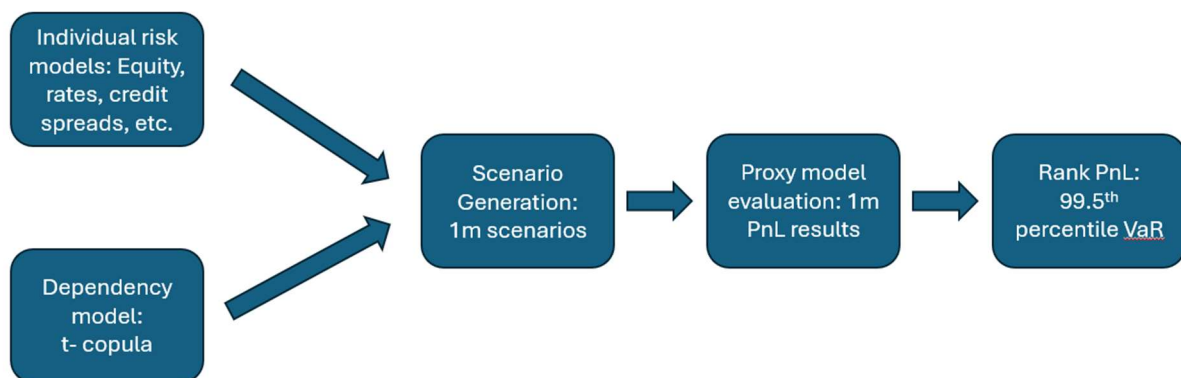
5. Classical vs Quantum Value at Risk

In their seminal work, Egger et al.[2] highlight the potential of the Quantum Amplitude Estimation (QAE) algorithm to achieve a quadratic speedups risk management applications, such as the calculation of Expected Value, Value at Risk (VaR), and Tail Value at Risk (TVaR). However, to fully appreciate the quantum advantage that QAE may offer, it is instructive to first understand the standard approach to Solvency Capital Requirement (SCR) calculations for Solvency II Internal Model (SII IM) firms without the use of quantum algorithms.

The conventional process of SCR calculation is a layered and computationally intensive task that involves several key components, which can be executed in parallel in a classical computing environment. In contrast, a quantum approach to VaR must consider the sequential nature of quantum operations and the necessity of maintaining quantum coherence throughout the computation.

Figure 4 Schematic of an SCR calculation:

Figure 4: Internal Model for SCR calculation



1. **Individual Risk Models:** These models quantify specific risks, such as equity, interest rates, and credit spreads. They form the basis of the scenario generation process.
2. **Dependency Model:** Typically employing a copula approach, this model captures the correlations and dependencies between different risk factors.

3. **Proxy Models:** These models simplify the relationship between risk factors and potential financial losses, enabling rapid computation of profit and loss (PnL) scenarios.

Together, the individual risk models and the dependency model comprise the scenario generation engine. This engine outputs a multitude of possible scenarios, which are then evaluated by proxy models to calculate the corresponding PnLs.

Although the high-level calculation framework is consistent across both classical and quantum computing platforms, the implementation diverges significantly due to the fundamental differences in information processing.

In classical computing, the problem is not encoded as a singular system. This allows for compartmentalization, where scenario generation and proxy model evaluation can occur independently.

In contrast, quantum computing inherently accommodates uncertainty and probabilistic outcomes, reflective of the quantum state's nature. However, to leverage this feature without disrupting the system's quantum state—specifically, to avoid collapsing the wave function—it is imperative to process the entire calculation within one quantum system. This involves evolving the state through a series of unitary operations that are synonymous with manipulating quantum information.

It is worth noting that gate-based quantum computers are theoretically capable of performing any computation that a classical computer can, classifying them as universal computers. Nonetheless, replicating classical computations on a quantum computer does not confer a quantum advantage. In fact, due to the slower 'clock speed' of quantum computers relative to their transistor-based classical counterparts, such an endeavour is bound to be much slower.

The quantum implementation requires a meticulous construction of the quantum state to represent the risk models and their dependencies. The implementation of quantum proxy models, while conceptually similar to classical models, necessitates a different approach to ensure the fidelity of the quantum state. The sequential nature of quantum operations also introduces new complexities in the computation process, as the entanglement must be carefully managed to preserve the correlations represented by the dependency model.

In the ensuing sections, we will delve into the intricacies of each component within the quantum framework, juxtaposing them against their classical analogues, and discuss the quantum advantage that may arise in the context of computational time, accuracy, and scalability. We will also consider the overhead time required to 'translate' classical problems into quantum-ready formats, a crucial factor in the overall efficiency of

quantum computing for financial applications that has not been sufficiently explored in prior discussions.

6. Quantum VaR

In this section, we explore a high-level framework for the adaptation of an Internal Model to quantum computing architecture. Initially, the focus will exclude the specific intricacies of the Quantum Amplitude Estimation (QAE) algorithm. This is a strategic exclusion, as the Internal Model's utility extends beyond merely calculating the 99.5th percentile Value at Risk (VaR); it is also utilized for generating the comprehensive distribution profile of Profit and Loss (PnL). So it is useful to know how a quantum computer can produce this information with or without a potential quantum advantage.

The QAE algorithm becomes particularly pertinent when the primary interest is centred on the Solvency Capital Requirement (SCR), which is a VaR calculation. This is especially the case when users intend to execute the computation multiple times across a range of initial conditions to thoroughly understand the sensitivity of their Capital Coverage Ratio (CCR).

Quantum circuits offer a compelling method for conceptualizing quantum computations, stripping away extraneous complexities and presenting the essence of quantum operations. For the development and visualization of quantum algorithms, I have employed the Qiskit framework, a Python-based library renowned for its utility in quantum programming. My implementations have been rigorously validated on both quantum simulators on my laptop, and those available through the IBM cloud platform (for larger problems).

The forthcoming discussions will provide a deeper dive into the methodology, including how each component of the Internal Model is re-envisioned in the quantum context, alongside a critical evaluation of the potential advantages and limitations inherent to quantum computing in this domain.

High Level Quantum IM calculation

Figure 5: Schematic for a Quantum Internal Model

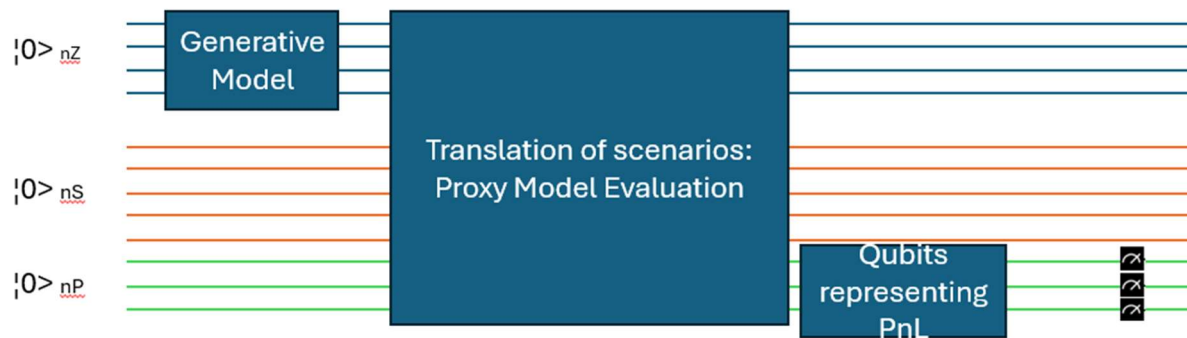


Figure 5 illustrates a conceptual schematic for implementing an Internal Model on a quantum computer, crucial for calculating the Solvency Capital Requirement (SCR) in financial risk assessment.

The schematic showcases a three-stage quantum process culminating in a measurement represented by 'meter' icons. Each stage corresponds to a crucial component of the Internal Model, executed using a unique set of qubits and quantum operations.

The first stage, denoted by the 'Generative Model' box, is implemented using nZ qubits. This generative model is designed to capture both the marginal risk distributions of individual risk factors and the complex dependencies between them. Encoding this information quantum mechanically can be approached in various ways. One method is an 'exact encoding,' which aims to represent the probability distribution of risk factors precisely. However, this method is computationally demanding, requiring an exponential number of quantum gates relative to the complexity of the risk model. Alternatively, one could employ a Quantum Circuit Born Machine (QCBM)[5], a quantum machine learning approach that utilizes the natural probabilistic nature of quantum mechanics to model distributions. While a QCBM may offer a more efficient encoding, it might not capture the risk distributions with the same precision as exact encoding. Each method has its trade-offs, with the exact encoding offering potentially greater accuracy at the cost of computational resources, while the QCBM provides a more resource-efficient solution, possibly at the expense of some fidelity in the risk distribution representation.

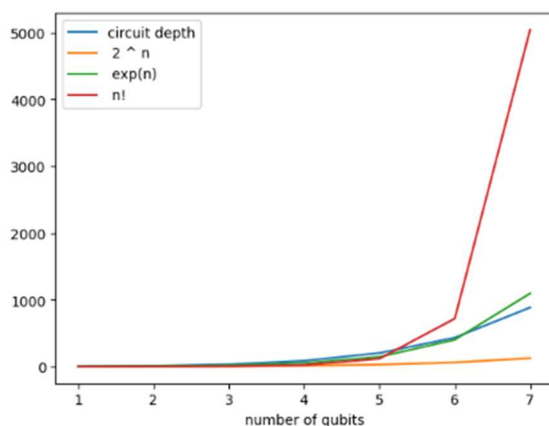
The subsequent box, labelled 'Translation of scenarios: Proxy Model Evaluation,' represents the translation of the risk scenarios generated by the generative model into financial metrics, typically PnL. This stage is where the quantum state resulting from the generative model is manipulated to reflect the impact of each scenario on the financial position. The number of qubits used here, nS , depends on the desired granularity of the

scenario analysis and the complexity of the proxy model. Sophisticated quantum algorithms are employed to evaluate the proxy models against each scenario efficiently. Finally, the qubits at the end of the schematic, nP , are designated for the output—representing the calculated PnL from the evaluated scenarios. The 'meter' icons indicate the measurement process, where the quantum information encoded in the qubits is translated back into classical information, which in this context refers to the PnL values for each scenario.

The entire process, from the generative model through to the measurement of PnL, is conducted within the quantum domain to maintain coherence and leverage the computational advantages of quantum mechanics. The aim is to utilize the inherent parallelism and probabilistic computations of quantum systems to perform risk analysis more efficiently than classical computers. The practical realization of this quantum Internal Model depends heavily on advances in quantum algorithms, error correction, and hardware capabilities. It is an area of active research, with the potential to revolutionize the field of financial risk assessment.

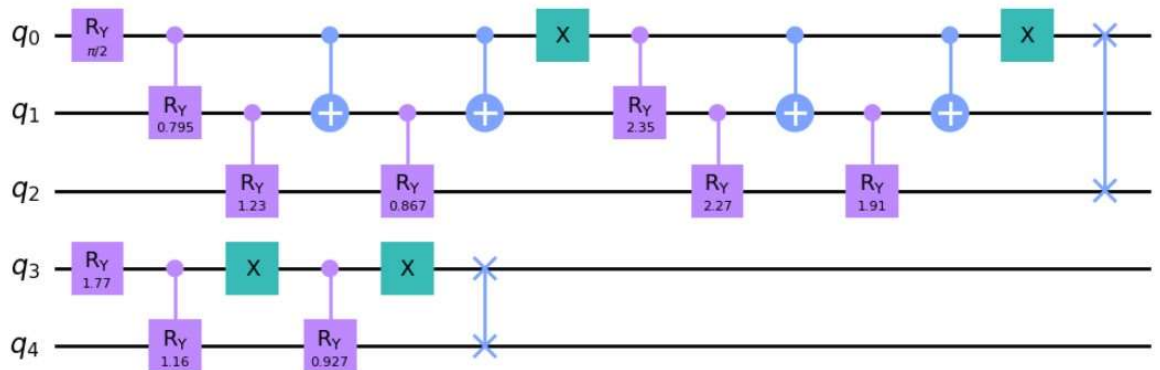
6.1 Quantum Generative Model

Figure 6: Qubits vs gate counts.



The figure above depicts the scaling of the circuit depth as a function of the number of qubits in the risk model. It illustrates that the circuit depth increases sharply, indicative of an exponential growth in the number of quantum gates required as more qubits are added. This aligns with the principle that an exact representation of the risk model necessitates a number of gates that grows exponentially with the number of qubits—this is because each qubit adds a dimension to the state space, and representing detailed distributions within this space becomes computationally heavier.

Figure 7: Exact implementation for 2 risks.



The figure above provides an example of an exact implementation for two independent risk factors encoded with 3(q_0, q_1 and q_2) and 2(q_3, q_4) qubits, respectively. This circuit diagram reveals the complexity and the specific quantum gates (such as the rotation gates R_x, R_y, R_z , and the controlled-NOT gates) required to prepare the state of each qubit accurately. The independence of the risk factors is reflected in the lack of entangling gates between the sets of qubits corresponding to each risk factor. The resulting probability distributions for the two risk factors are shown below:

Figure 8: Histograms for the encoded risk factors.

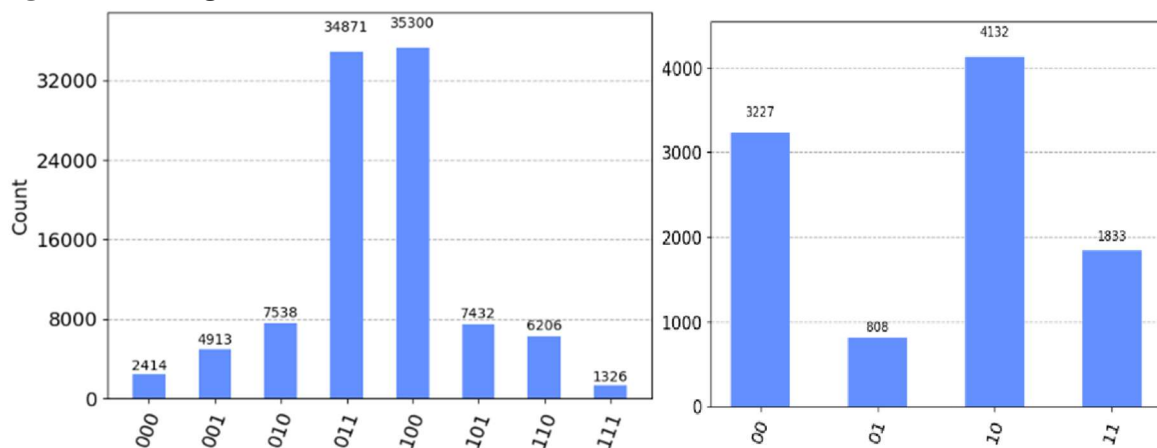


Figure 9: Quantum Circuit Born Machine (QCBM) Architecture

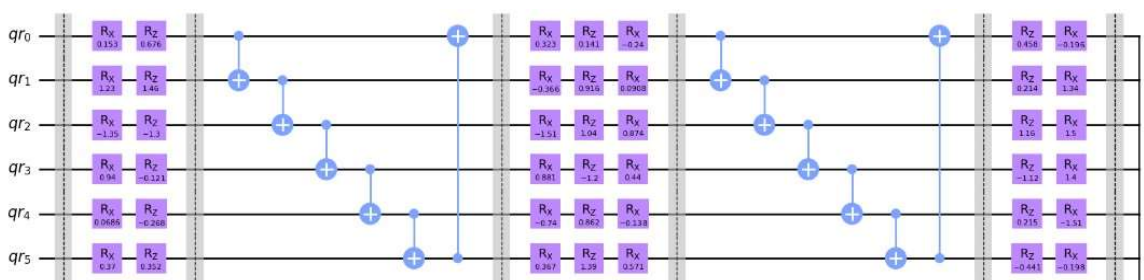


Figure 9 demonstrates the architecture of a QCBM, which has a fixed gate count regardless of the number of qubits. This QCBM comprises layers of parameterized rotation gates (Ry) and entangling operations (CX gates). The fixed architecture suggests a more scalable approach compared to the exact implementation. The parameters of the rotation gates in the QCBM are adjusted during a training phase, where the circuit learns to approximate the probability distributions of risk factors.

We observe that the exact implementation, while precise, is less scalable due to its exponential increase in gate count. This poses a significant limitation for modelling complex risk factors as quantum hardware is currently limited in terms of qubit count and gate fidelity. On the contrary, the QCBM approach is a more scalable solution with a fixed number of gates that only increases linearly with the number of qubits, making it a potentially more practical choice for larger-scale problems.

The exact implementation would require significant quantum resources to accurately capture complex risk distributions and their dependencies, which could be a limiting factor for near-term quantum devices. The QCBM, while it may not capture the full complexity of the risk distributions, offers a trade-off between accuracy and feasibility on current quantum hardware. The QCBM does however, require 'training' which is an overhead computationally. From a practical standpoint, the overhead might be acceptable if it allows one to run the problem on near term quantum hardware by significantly reducing the 'circuit depth' of the problem.

Other approaches from the field of Quantum Machine Learning (QML) are also available, for example, similar to the QCBM, Quantum Generative Adversarial Networks (QGANs) can also be 'trained' to create a generative model, with a different architecture, and trade-offs.

In summary, the choice between an exact implementation and a QCBM will be guided by the trade-offs between the precision of risk modelling and the scalability of the quantum hardware available. The exact approach is resource-intensive but precise, suitable for small-scale models, while the QCBM(or QGANs) offers a pragmatic approach for larger-scale applications where hardware limitations are a concern.

6.2 Quantum Proxy Representation

The adoption of proxy models by insurance firms for SII IM purposes, typically in the form of polynomials, is driven by their analytical tractability and computational expedience. However, the intricacies of these models can vary substantially. The complexity is often a reflection of the multifaceted nature of the insurer's balance sheet, which can encompass a wide array of risk factors, geographical diversity, and the intricacies of calibration algorithms tailored to these conditions.

The proxy model I have chosen for illustration is far from trivial, encompassing a substantial 836 terms to model the Net Asset Value (NAV) directly, key features are summarised below:

# of terms	836
# of risks	31
Maximum # of risks for interactions	5
Maximum order of individual risk terms	4
Maximum order for interaction terms	3

This level of complexity stands in stark contrast to the simpler models seen in foundational quantum finance works by Egger et al. For instance, in their 2019 paper on Quantum Risk Management, published in Quantum Finance, Egger et al. detail a two-term polynomial of first order to express risk [2]. Their model, denoted by $F(x,y)=1-0.1349x-0.0186y$, not only epitomizes simplicity but also presents a clear monotonic decrease with respect to the designated computational basis states.

Moreover, Egger et al.'s subsequent 2020 exploration, featured in IEEE Transactions on Quantum Engineering, touches upon the modelling of a portfolio's loss due to credit defaults, where losses given default (LGD) are exclusively positive integers [3]. This scenario conveniently circumvents the need for any polynomial mappings across both negative and positive ranges, thus simplifying the translation to quantum basis states defined over a purely positive interval $[0, 2^n - 1]$.

In stark contrast, the polynomial proxy model I am working with necessitates a nuanced quantum computational strategy. This is not just due to the sheer volume of terms but also due to the multidimensional interactions and the high orders of individual risk terms. Here's a more detailed look at the quantum intricacies entailed:

- 1) Each risk factor within the model is accorded a quantum representation using a set of 20 qubits. This is to ensure a granularity that approximates the continuous nature of risk variables, thereby avoiding the pitfall of

discretization. Such granulation enables us to model each risk variable over more than a million discrete states.

- 2) The methodology for the quantum evaluation of such a proxy model requires us to substitute mapped qubits for each risk factor, finely tune the coefficients for all polynomial terms, and apply sophisticated quantum algorithms for 'Weighted Addition' to compute the collective impact of each term.

Taking the RPI_PC1 risk factor as an exemplar, which signifies a parallel shift in the Retail Price Index (RPI) curve, we encode this using 20 qubits, effectively capturing a normal distribution within the bounds of a practical range. The process involves translating this normal distribution onto a quantum scale, ranging from 0 to 1,048,575 in quantum basis states.

The interactions between multiple risk factors requires entangling qubits, and further amplifies the complexity. As highlighted, a simple interaction between two such factors could result in hundreds of new terms, each necessitating quantum representation.

In terms of quantum resource allocation, the requirement is prodigious. Each risk factor, encoded with 20 qubits, generates a series of states that must be multiplied with a proxy coefficients, the same is true for higher powers of the risk factors.

When it comes to interactions, the requirements multiply. A single interaction term between two risk factors, each depicted by 20 qubits, translates to 400 potential interactions—each needing quantum representation.

When extrapolated to the full scale of the model with 31 risk factors, the interactions become exceedingly complex.

For the model at hand, with 836 terms including multi-risk cross terms, the quantum computational load is immense. The exact quantum representation would necessitate a very large number of qubits, running into the tens of millions. Specifically, just the affine map to go from a proxy model expressed in terms of risk factors, to one expressed in terms of qubits would require a staggering 35,247,385 qubits. I should note that the qubit requirement depends on the exact nature of the proxy model, and that capturing the coefficients and the weighted sums requires further qubits, but that requirement is marginal compared to the above.

This quantum proxy model, hence, extends far beyond the models examined by Egger et al. in their seminal works on quantum finance [2] [3]. The models they proposed are significantly less demanding in terms of quantum resources (handful of qubits), often involving simpler functions with fewer terms and lower-order interactions. By contrast, the quantum representation of the proxy model I propose reflects a high-dimensional

and multifaceted risk landscape that would benefit from the advanced capabilities of quantum computing.

The transition from a classical polynomial expression to its quantum counterpart involves a meticulous process of encoding continuous distributions into discrete quantum states and translating interaction terms into entangled states that accurately reflect the interdependencies among risks.

6.2.1 Polynomial to Qubit representation for a simple case

The translation from a polynomial expression in classical risk factors to a quantum representation in terms of qubits is an intricate process that benefits from examination through a simpler example. Let's consider the following polynomial function:

$$F(x,y) = 2x - xy + 3y^2$$

Here x and y are risk factors represented by 2 qubits each $x:[q_0, q_1]$, $y:[q_2, q_3]$

To further simplify, we assume that x and y are uniformly distributed within the interval [0, 3], aligning precisely with the basis states producible by pairs of qubits. The relationship between the qubits and the risk factors is given by:

$$x = q_0 + 2q_1$$

$$y = q_2 + 2q_3$$

Substituting these relationships into the polynomial F yields:

$$\begin{aligned} F(q_0, q_1, q_2, q_3) &= 2(q_0 + 2q_1) - (q_0 + 2q_1)(q_2 + 2q_3) + 3(q_2 + 2q_3)^2 \\ &= 2q_0 + 4q_1 - q_0q_2 - 2q_0q_3 - 2q_1q_2 - 4q_1q_3 + 3(q_2^2 + 4q_2q_3 + 4q_3^2) \end{aligned}$$

When dealing with qubits, we apply the principle that the square of a qubit state is the state itself, simplifying qx^2 to qx . Thus, the polynomial in terms of qubits is:

$$\begin{aligned} F(q_0, q_1, q_2, q_3) &= 2q_0 + 4q_1 - q_0q_2 - 2q_0q_3 - 2q_1q_2 - 4q_1q_3 \\ &\quad + 3q_2 + 12q_2q_3 + 12q_3 \\ &= 2q_0 + 4q_1 + 3q_2 + 12q_3 - q_0q_2 - 2q_0q_3 - 2q_1q_2 - 4q_1q_3 + 12q_2q_3 \end{aligned}$$

This simplification leads us to a quantum expression for F that involves a total of 9 terms, which is an increase from the original 3 terms in the classical polynomial. It is important to note that each term in the qubit representation carries a coefficient that is

derived from the original polynomial function, and these coefficients are fundamentally linked to the number of qubits that represent each risk factor.

To effectively translate classical risk factor distributions into their quantum analogue, the range of each risk factor must be carefully considered to ensure proper qubit encoding. In the earlier example, we benefited from the uniform distribution of risk factors x and y , which coincided neatly with the qubit basis states, allowing for a direct substitution. However, when dealing with normally distributed risk factors with a mean of 0, the conversion demands a more nuanced approach to accommodate the symmetric distribution around the mean.

For instance, if x and y are normally distributed within the range $[-3,3]$, the mapping to qubit representation requires an adjustment to reflect this range. The translation would involve scaling and translating the qubits' basis states to the continuous risk factor values. The mapping might take the following form:

$$X = -3 + (q_0 + 2q_1) / (3 - (-3)) * (3 - (-3))$$

Here, the factor $(3 - (-3)) / (3 - 0)$ in the denominator serves to normalize the range of the qubit representation, scaling it to the range of the risk factor. The numerator adjusts the qubits' values within the desired range. The constants 3 and 0 in the denominator represent the highest and lowest values in the qubit basis states, corresponding to the binary values of q_0 and q_1 (where '00' maps to 0 and '11' maps to 3, for two qubits). The addition of -3 is necessary to shift the scaled qubit values to centre around the mean of the normal distribution.

In practice, when both q_0 and q_1 are in the '0' state, the mapping yields the lowest point of the range, -3. Conversely, when both are in the '1' state, it corresponds to the highest point, 3. This mapping ensures that the full spectrum of the normal distribution is accounted for within the qubit representation.

This conversion process is crucial for aligning the quantum computational framework with the probability distributions intrinsic to the risk factors. It exemplifies the detailed attention required to maintain fidelity between the classical statistical models of risk and their quantum counterparts. Such transformations are foundational to accurately encoding and processing financial models on quantum platforms.

6.2.2 Capturing Coefficients

The task of calculating weighted sums of qubit terms is essential for the evaluation of complex financial models, such as the proxy models used in quantum risk analysis. In the quantum computing framework, particularly with frameworks such as Qiskit, this operation is facilitated by algorithms like the `WeightedAdder`. However, due to the nature of this method, which is designed to handle positive integers, the representation and handling of qubit coefficients require special attention.

Let's elaborate on how to navigate these nuances:

Handling Real Coefficients with Weighted Adder: The `WeightedAdder` algorithm operates with integers, presenting a challenge when working with real-numbered coefficients, which are common in financial models. To reconcile this:

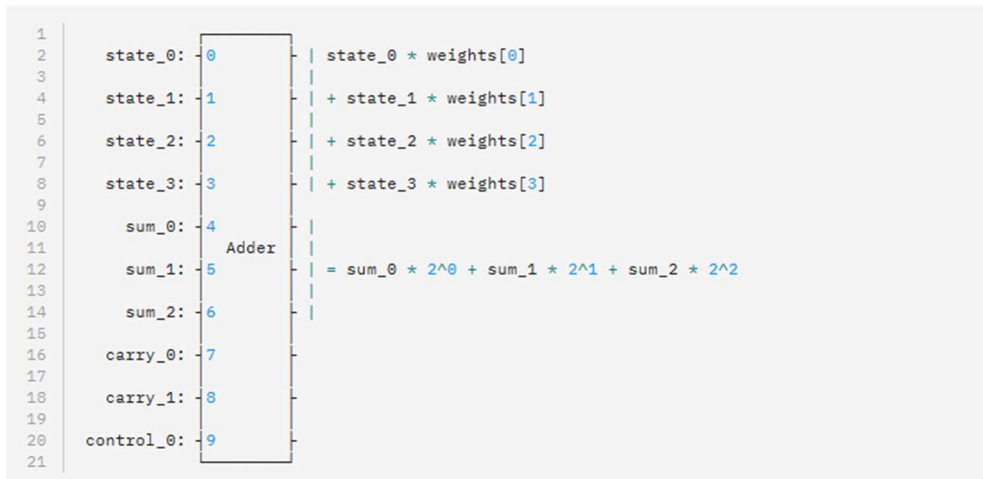
- Scaling to Integers:** By scaling each coefficient by a factor of 10^n , where n is chosen so that all the coefficients are transformed into integers, we can preserve the precision of the coefficients. For instance, if we have coefficients accurate to 16 decimal places, a scaling factor of 10^{16} will convert all coefficients to integers without loss of precision. This method is advantageous because it is straightforward and maintains the granularity required for accurate financial modelling.
- Decimal Encoding:** Alternatively, coefficients could be encoded as decimals within the quantum computing framework.

The scaling approach is preferred due to its simplicity and reliability, allowing for high precision while ensuring compatibility with the `WeightedAdder`'s requirements.

Managing Positive and Negative Weights: Since the `WeightedAdder` is designed for positive weights, the calculation needs to be divided into distinct operations to handle positive and negative weights separately, combining them at the end:

- Sum of Positive Weights:** Aggregate all qubit terms with positive coefficients, applying the `WeightedAdder` to sum these values.
- Sum of Negative Weights:** Separately, sum all qubit terms associated with negative coefficients. This involves negating the coefficients temporarily to utilize the `WeightedAdder`.
- Combination of Sums:** Finally, combine the results from the positive and negative sums. This can be achieved using Two's complement, which is a mathematical operation on binary numbers that effectively handles subtraction of binary numbers as an addition.

Figure 10: Weighted Adder schematic.



Source: [WeightedAdder | IBM Quantum Documentation](#)

Resource Requirements for Weighted Additions: While the WeightedAdder method introduces additional qubits to hold the output, the number required is relatively small compared to the large-scale multi-qubit interactions represented by the c35m qubits. However, it's crucial to allocate sufficient qubits to store the interim and final results of the weighted additions, especially when dealing with high-precision calculations as necessitated by complex financial models.

In practice, the implementation of these steps must be carefully designed to ensure that the quantum circuit can handle the computational load and that the encoded polynomial is accurately reflected in the quantum state. With precision and careful construction, the quantum algorithm for weighted addition becomes a powerful tool for financial analysis, opening the door to more efficient and potentially more insightful risk assessments.

Additional qubits will be required to 'hold' the output of the 'weighted additions', but the requirement is marginal compared to the c35m for multi qubit interactions.

The output from the positive and negative weighted additions can be summed by using Two's complement.

6.2.3 Calculating Net Asset Value

The depiction of the Weighted Adder circuit in Figure 6 elucidates the structure of the quantum weighted sum, revealing the precise quantity of qubits necessary for this task. The circuit's design allows for the computation of the maximum possible weighted sum, achieved when all contributing qubits are in the state $|1\rangle$. This maximum sum can be mathematically expressed as:

$$S = 1 + \lfloor \log_2(\sum_{j=0}^{n-1} \lambda_j) \rfloor$$

Here, the floor function $\lfloor \ \rfloor$ represents rounding down to the nearest whole number, and λ_j is the weight assigned to the j^{th} qubit.

Subsequent to the computation of weighted additions, it becomes necessary to amalgamate the two 'sum registers'—one for the weighted sum derived from positive weights and another from negative weights.

6.2.3.1 Two's complement

Due to their distinct representations of positive and negative sums, a direct addition is not feasible. To circumvent this, each is converted into the 'Two's Complement' format, enabling their subsequent combination.

Two's complement is a mathematical operation and binary encoding scheme for representing both positive and negative integers in binary number systems. It's the most common method of representing signed integers on computers.

For positive numbers the two's complement representation is the same as the standard binary representation. For negative numbers, we start with the binary representation of its absolute value, flip all the bits and add 1 to the least significant bit.

For example, to represent -5 in a two's complement system with at least three bits:

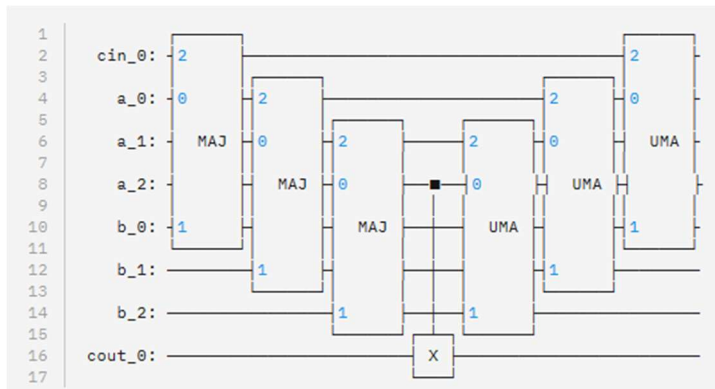
- Start with the binary representation of 5, which is 101.
- Invert the bits to get the one's complement: 010.
- Add 1 to the LSB to get the two's complement: 011, so -5 is represented as 1011 in a four-bit system. Notice the extra 1 that gets appended as the MSB for a negative number in this format, its positive counterpart would be 0101.

Two's complement greatly simplifies the process of binary subtraction, as it allows for the use of the same addition circuits as addition. For example, subtracting a number is equivalent to adding its two's complement, adding 5 and -5 would yield $1011 + 0101 = 0000$, as expected.

6.2.3.2 Ripple Carry Addition

For the amalgamation of these two registers (post Two's complement), the CDKM RippleCarryAdder can be utilized effectively. This adder stipulates that both 'sum registers' match in size, necessitating the expansion of the smaller register with additional qubits to ensure compatibility.

Figure 11: Schematic of CDKM Ripple Carry Adder



Source: [CDKM Ripple Carry Adder | IBM Quantum Documentation](#)

In **Figure 11**, the CDKM Ripple Carry Adder's configuration is illustrated, outlining the fundamental structure of ripple carry addition. Here, cin_0 symbolizes the initial carry-in qubit, a_0, a_1, a_2 the qubits of the first register, and b_0, b_1, b_2 those of the second register. The carry-out from the addition is denoted by cout_0.

The resultant sum through this addition process is effectively captured by four qubits, represented as $|cout_0, b_2, b_1, b_0\rangle$. This showcases the quantum circuit's ability to compute and represent the weighted sum of inputs, highlighting the intricacies involved in translating classical computation methods to the quantum realm and illustrating the practical steps taken to address these challenges within a quantum computing context.

6.2.4 Calculating VaR

Upon completion of the preparatory steps, executing the quantum circuit multiple times serves a purpose similar to running simulations in a Monte Carlo method. However, a distinctive advantage in the quantum scenario is the generation of truly random samples, as opposed to the pseudo-random nature of samples in classical computing. The exact number of iterations or "shots" required for the circuit will vary depending on the specific convergence needs of the Solvency Capital Requirement (SCR) calculation. Insurance and financial institutions often rely on a range of 500,000 to 1,000,000 simulations to ensure convergence of their SCR estimates.

The result of this quantum sampling process yields a distribution reflecting the Net Asset Value (NAV) of a company. This distribution is the outcome of the encoded proxy model—a polynomial function mapping the myriads of risk variables such as interest rates, inflation rates, property values, equity prices, and longevity risks, directly to the company's value.

Importantly, the output states from the quantum circuit are presented in a Two's complement format. This encoding facilitates the handling of both positive and negative values within the binary system, making it a standard approach for representing signed numbers in computing. To derive meaningful NAV values from each scenario, these Two's complement-encoded outputs must be decoded back into their respective integer values. This decoding step is crucial for translating the binary data produced by the quantum circuit into actionable financial insights, enabling analysts to evaluate the potential impacts of various risk scenarios on the company's NAV. Finally, we must divide the resulting NAVs with the number used to scale the qubit representation coefficients.

This process underscores the potential of quantum computing to revolutionize financial modelling and risk assessment, offering a new dimension of randomness and computational efficiency. Yet, it also highlights the need for specialized post-processing—such as the decoding of Two's complement outputs—to bridge the gap between quantum computation results and practical financial analysis.

6.3 VaR using Quantum Amplitude Estimation

Having explored the direct computation of Value at Risk (VaR) with quantum computing, we now focus on quantum algorithms that confer a quantum advantage, particularly through Quantum Amplitude Estimation (QAE).

QAE, as delineated in the foundational work [1], is pivotal for achieving nearly quadratic acceleration for VaR applications. It leverages an operator A to initiate a superposition across $n+1$ qubits such that it results in the final qubit having a probability a of being observed in state $|1\rangle$. The overall structure of the problem can be written down as follows:

$$A|0\rangle_{n+1} = \sqrt{1-a} |\Psi_0\rangle_n |0\rangle + \sqrt{a} |\Psi_1\rangle_n |1\rangle$$

The operator A acts on $n+1$ qubits initialised to $|0\rangle$, creates a super position, such that the last qubit will upon measurement find itself in state $|1\rangle$ with probability a .

The goal is to estimate the VaR of a portfolio using quantum amplitude estimation. To do this, we need to construct a quantum operator A that can encode the portfolio loss into a quantum state.

A takes $n+1$ qubits in the state $|0\rangle$ and creates a superposition of states, where the last qubit depends on the value of the portfolio loss. If the portfolio loss is greater than a certain threshold, the last qubit will be in the state $|1\rangle$ with probability a . Otherwise, it will be in the state $|0\rangle$. The threshold is chosen to match the desired confidence level for the VaR calculation.

For example, for the SCR, we want $a=0.5\%$, which means that the last qubit will be in the state $|1\rangle$ with 0.5% probability if the portfolio loss exceeds the SCR.

By implementing a bisection search, we can find the VaR in at most m steps, where m represents the number of qubits required to encode the PnL (the number of qubits in sum register for CDKM Ripple Carry Addition).

To summarise, our strategy to utilise QAE for SCR calculation is as follows:

- 1) Encode the individual risk models and the dependency structure. This is such that each of the individual risks are distributed over their computational basis states $|0\rangle$ to $|2^{n_i}-1\rangle$ where n_i represents the number of qubits encoding risk i .
- 2) The proxy model capturing the sensitivity of the Own Funds (Net Asset Value) will be 'reformulated' such that it aligns to the qubit basis states for each of the risks, and interactions.
- 3) Implement a 'comparator' such that it will flip the state of the last qubit to $|1\rangle$ if the loss exceeds a particular threshold.
- 4) Estimate the probability a of the measurement qubit finding itself in state $|1\rangle$
- 5) Bisection search until ' a ' equals the desired confidence level for the VaR calculation.

6.3.1 But where is the quantum advantage?

We started by setting out that there is a quadratic speedup when using QAE, but have ended up specifying an algorithm that requires repeating the calculation up to m times. Whilst superficially this does not look more efficient, it is theoretically more efficient than calculating the entire PnL distribution directly, as set out below:

- 1) Measuring just the indicator vs measuring all qubits representing the PnL
- 2) Fewer 'shots' required to measure the probability associated with a particular threshold. For example, if we are interested in the 99.5th percentile (1 in 200), running 1000 shots might be sufficient to estimate the pvalue of a particular threshold accurately. We know through [2] that the standard error using QAE is $1/n$, as opposed to $1/\sqrt{n}$. So, in theory this method requires up to $m*1000$ shots, which is considerably less than 1,000,000 typically used, even for $m=60$.

6.3.2 Building a Comparator

The job of the comparator is to flip the state of the ‘measurement’ qubit, if the PnL exceeds a specified threshold.

As the PnL output will be encoded in a binary format, we can utilise standard binary arithmetic techniques to accomplish this task.

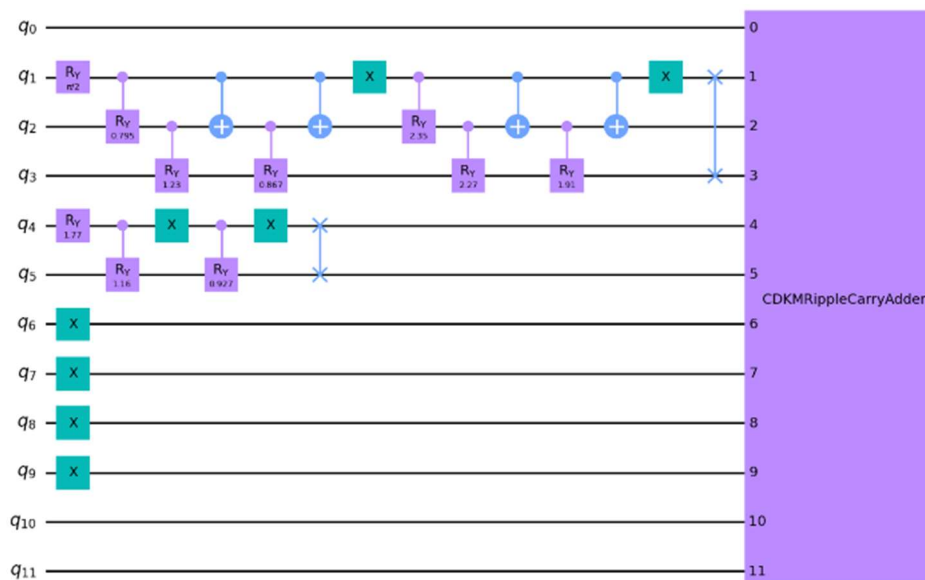
Once again, the Two’s complement format is used to achieve the required transformation (i.e. a carry on the extra qubit when the threshold is exceeded).

To recap, Two’s complement format allows us to add positive and negative numbers without having to worry about signs. In the context of building a comparator, a ‘carry out’ is required if the PnL exceeds a certain pre-specified threshold.

This can be achieved by

- 1) Encoding the two’s complement of the threshold
- 2) Adding the above to the PnL
- 3) If the PnL exceeds the threshold, there will be a carry out on the ‘measurement qubit’.

Figure 12: Stylised example of a comparator circuit.



The above figure shows the implementation of a comparator for the quantum generative risk models shown in **Figure 7**.

Here the PnL of the system is simply the computational basis state obtained by measuring the qubits $|q_5q_4q_3q_2q_1\rangle$ converted into an integer.

The Two's complement of the threshold is encoded in qubits $|q_{10}q_9q_8q_7q_6\rangle$, which is 01111(in the above example), which means that the threshold is a positive number 10001, i.e. 17. The qubit q_0 , and q_{11} represent the carry in and carry out qubits respectively.

This implementation will produce a carry out in qubit q_{11} if and only if the state $|q_5q_4q_3q_2q_1\rangle$ exceeds 17, i.e. when q_5 and at least one other qubit is in state $|1\rangle$.

6.3.2 Encoding the threshold.

To recap, QAE offers a sophisticated means to accurately estimate the probability that a measurement qubit will be found in the state $|1\rangle$. For VaR calculations, we aim to determine the 'PnL', such that when it is used as a 'threshold', the probability of observing the 'measurement qubit' in state $|1\rangle$ is equal to the confidence level for the VaR.

In this process, we construct a quantum operator, referred to as A, that is designed to flip the state of the measurement qubit to $|1\rangle$ only when the simulated financial loss surpasses this threshold. Importantly, the actual value of the loss is not directly observed; instead, its effect is inferred through the qubit's state.

A notable challenge arises from the encoding of PnL values using the two's complement system. This binary representation allows for the encoding of both positive and negative values but results in a non-linear sequence of numbers, due to the fact that negative numbers have their most significant bit (MSB) set to 1, contrasting with 0 for positive numbers. This discrepancy disrupts the monotonic progression required for the comparator to function accurately, as demonstrated in the table below.

Integer	Two's complement
-5	1011
-4	1100
-3	1101
-2	1110
-1	1111
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101

When setting a threshold, such as 2, the binary representation of this value and those above it should ideally result in the measurement qubit flipping to $|1\rangle$. However, due to

the non-monotonic ordering of two's complement values, both negative and positive PnL values around this threshold would incorrectly trigger the flip. Specifically, PnL values of -5 to -1, alongside 2, 3,4 and 5, would all cause the qubit to flip, despite our intention to target only positive values exceeding the threshold.

To resolve this, we introduce an innovative approach by adding an extra bit ahead of the MSB to explicitly denote the sign of the number. This adjustment ensures a monotonically increasing sequence of two's complement representations, directly aligning with the integers they represent. Consequently, the binary sequence now facilitates an accurate comparison, allowing the comparator to precisely identify when the PnL exceeds the set threshold.

Integer	modified Two's complement
-5	01011
-4	01100
-3	01101
-2	01110
-1	01111
0	10000
1	10001
2	10010
3	10011
4	10100
5	10101

With this modification, the new threshold for the comparator is correctly set to the binary equivalent of 2 (10010) in our modified notation, ensuring that the measurement qubit is flipped to $|1\rangle$ only under the appropriate conditions.

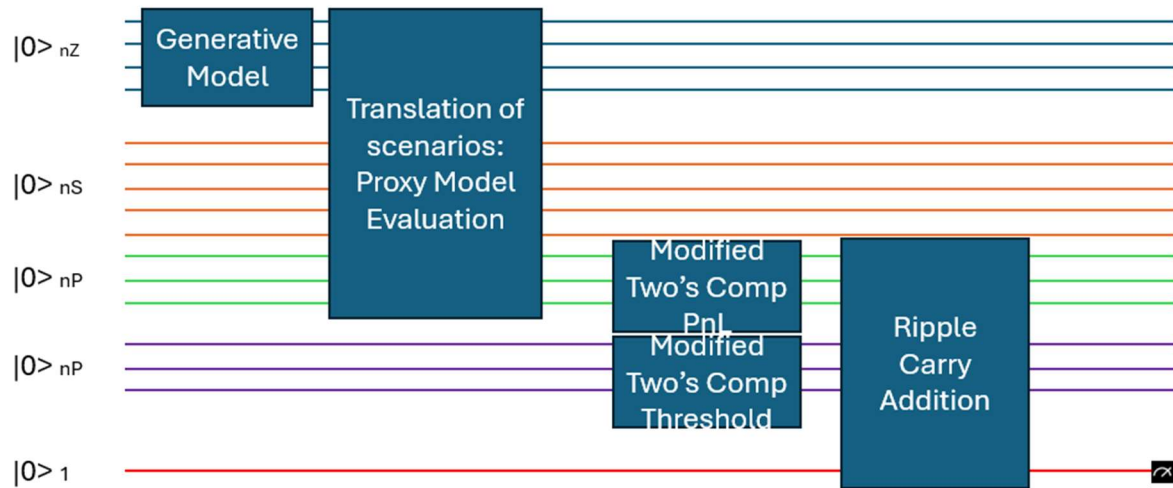
Note that the state of the Most Significant Bit(MSB) depends on whether the PnL is positive or negative. This does not require that we explicitly measure the qubits representing the PnL, as described in **6.2.4 Calculating VaR**. Instead, we can use a controlled-Not gate to flip the state of the extra qubit to $|1\rangle$ if the MSB is $|0\rangle$, or keep the extra qubit at $|0\rangle$ otherwise.

6.3.3 How to Two's complement an 'un-observed' quantity.

As mentioned above, the QAE based approach is different to the Montecarlo based approach where an explicit distribution of PnL is generated. In the QAE case, we ensure that all the risk models, and their relationships to each other and the PnL are encoded in the quantum system, however, directly measuring the PnL could result in a 'collapse of the quantum state' and any quantum advantage would be lost. To preserve the structure of the problem in the quantum state, we append a 'comparator circuit' which

would flip the state of a ‘measurement qubit’ initialised to $|0\rangle$ if the PnL exceeds a prespecified threshold, we illustrate below.

Figure 13: Building the operator A for QAE



The figure above illustrates how the operator A, in the context of QAE is implemented.

Notice, that in contrast to **Figure 5** there are two further circuits that are implemented, one for encoding the threshold, and the other to compare the PnL to the Threshold. Finally, only the last qubit is measured.

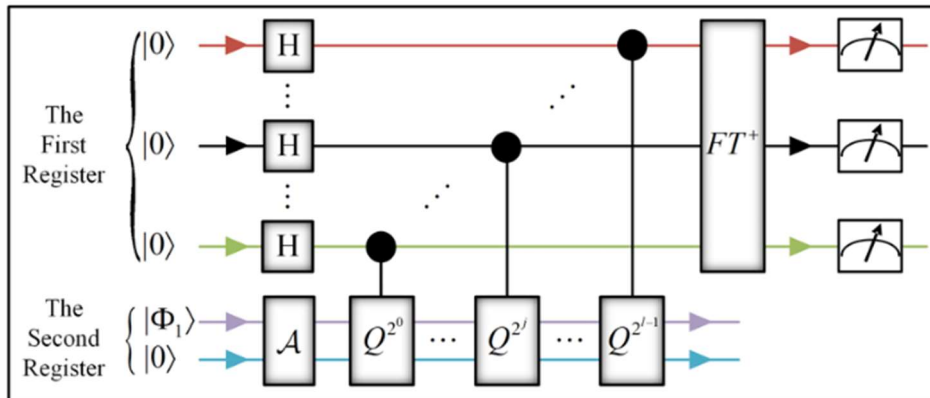
In practice, the last qubit is not measured, as we use QAE to estimate the probability of finding this qubit in state $|1\rangle$, but it is instructive to show the calculation in this manner.

6.3.4 Quantum Amplitude Estimation.

Now that we have a strategy for implementing the operator A, let's dive into the QAE algorithm itself.

QAE leverages the quantum effect of ‘phase kickback’ and a mathematical operation called Inverse Quantum Fourier Transform to efficiently estimate this probability a, that the ‘objective qubit’ will find itself in state $|1\rangle$. The accuracy of the estimate depends on the number of qubits used for sampling the ‘phase’.

Figure 14: Canonical Quantum Amplitude Estimation



Source: [The quantum circuit of the generalized quantum amplitude estimation | Download Scientific Diagram \(researchgate.net\)](#)

The figure above shows the schematic for the canonical QAE algorithm[7]. This algorithm has the following distinct components:

The A operator, which we have already covered.

The measurement register (First Register in the diagram above), which is prepared in an equal superposition. This moves the individual qubit states to the ‘equator’, allowing a phase to be encoded at a later stage through ‘phase kickback’.

The controlled Q operators, generate a phase kickback into the measurement register. These operations encode the binary representation of the probability amplitude as phases in the **measurement register**.

The inverse QFT, which makes the ‘phase’ an observable quantity upon measurement.

Post-processing, the measured quantity will not be probability of the ‘objective qubit’ being in state $|1\rangle$, rather it will be the probability amplitude encoded as a phase into the ‘First Register’. Therefore, the phase information needs to be decoded back into a probability.

The measurement outcome is related to probability via the following formula:

$$P(\phi, N) = \sin^2 \left(\frac{\phi}{2^N} \right) \cdot \frac{\pi}{2}$$

Φ is the observation in binary translated into an integer,

N is the number of qubits in the First register

Increasing the number of measurement qubits increases the granularity with which the phase can be measured, which in turn increases the resolution of the probability grid for our VaR calculation. However, given the above function is non-linear in ϕ , we find the resolution of the probability grid is much higher at the edges compared to the centre. We have shown this effect for N=8 qubits below.

binary	Integer	Probability	difference
00000000	0	0.0000%	
00000001	1	0.0038%	0.0038%
00000010	2	0.0151%	0.0113%
00000011	3	0.0339%	0.0188%
00000100	4	0.0602%	0.0263%
00000101	5	0.0941%	0.0339%
00000110	6	0.1355%	0.0414%
00000111	7	0.1844%	0.0489%
00001000	8	0.2408%	0.0564%
00001001	9	0.3047%	0.0639%
00001010	10	0.3760%	0.0714%
00001011	11	0.4549%	0.0788%
00001100	12	0.5412%	0.0863%
00001101	13	0.6349%	0.0938%
01111000	120	45.0991%	0.6103%
01111001	121	45.7101%	0.6110%
01111010	122	46.3218%	0.6116%
01111011	123	46.9340%	0.6122%
01111100	124	47.5466%	0.6127%
01111101	125	48.1596%	0.6130%
01111110	126	48.7729%	0.6133%
01111111	127	49.3864%	0.6135%
10000000	128	50.0000%	0.6136%
11110100	244	99.4588%	0.0938%
11110101	245	99.5451%	0.0863%
11110110	246	99.6240%	0.0788%
11110111	247	99.6953%	0.0714%
11111000	248	99.7592%	0.0639%
11111001	249	99.8156%	0.0564%
11111010	250	99.8645%	0.0489%
11111011	251	99.9059%	0.0414%
11111100	252	99.9398%	0.0339%
11111101	253	99.9661%	0.0263%
11111110	254	99.9849%	0.0188%
11111111	255	99.9962%	0.0113%

Note we get sufficient resolution at the edges for a 99.5th percentile VaR calculation. If the PnL function represents losses as a positive value, then the corresponding percentile will be 99.5%, otherwise 0.5%.

The reader might be intrigued why we have estimated the probability in such a round about way, going through phases and trigonometric functions, rather than finding a way of encoding the 'a' directly as a binary number that can be extracted upon measurement. The power of QAE stems from its ability to leverage 'interference patterns', particularly in cases where the probabilities are not known a priori and cannot be directly encoded.

Above, we have presented the canonical Amplitude Estimation algorithm, however, due to the repeated controlled-Q operations, it requires large circuits and is computationally expensive. There are other faster variations available, such as Iterative Amplitude Estimation [8], Maximum Likelihood Amplitude Estimation [9], and Faster Amplitude Estimation [10].

All the above approaches are available in Qiskit via the **AmplitudeEstimator** Interface.

6.3.5 Bringing it all together: Binary search.

The section above describes how the probability amplitude associated with finding the objective qubit in state $|1\rangle$ is encoded as the phase in the First Register. There are a total of 2^N basis states corresponding to different probabilities/phases, which can be quite a large space to explore. With binary search/bisection search, we can find Threshold, that results in a probability estimate in line with the VaR confidence interval in at-most N steps.

The quantity we will vary in each iteration of the bisection search will be the PnL Threshold, which we saw in the section on Comparators. The Threshold that corresponds to the probability of 0.995 will be the SCR, i.e. loss that exceeds 99.5th percentile of the loss distribution.

Whilst the standard bisection search algorithm assumes that the ends of the maximum and the minimum values of the PnL are known a priori, this will not necessarily be the case for the SCR calculations. We sample the PnL directly as in [Section 6.2.4](#) but that will defeat the purpose of using QAE.

Even though we do not know the bounds of our PnL function, we do know that the losses will be sorted, thanks to the Adjusted Two's complement approach in [Section 6.3.2](#).

Thanks to SCR calculations from a previous cycle, we will always have reasonable guesses available for the bounds, we can use the following approach to establish updated bounds before kicking off the bisection search.

1. **Start with an initial guess** for the bounds. This can be the maximum and minimum PnL from a previous SCR calculation.

2. **Check if the target is outside the guessed bounds.** If the target is higher than the upper bound, increase the upper bound exponentially (e.g. double it) and repeat the check. If the target is less than the lower bound (in cases where the minimum is not known), decrease the lower bound exponentially. We can check whether the target is inside or outside the bounds by running the QAE algorithm with the Threshold set as each of the bounds. If the $[\min, \max]$ produce probabilities $[0.1, 0.9]$ we know that the target value is outside bounds, as we are interested in $[0.005]$, in this case we would exponentially decrease the lower bound.
3. **Repeat this process** until the target value is within your guessed bounds.
4. Once suitable bounds are found, **apply the bisection search** within these bounds to find the target value.

7. Quantum Advantage

Quantum advantage represents a crucial milestone in the evolution of quantum computing, signifying the juncture at which a quantum computer outperforms classical computers at a specific task, achieving results faster or more efficiently than any existing classical system could under practical conditions. This concept does not imply that quantum computers are universally superior to classical computers; rather, it highlights their superior performance in handling certain specialized tasks that are inherently complex for classical computing architectures.

Such tasks might include complex simulations of quantum mechanical systems, optimization problems in logistics and manufacturing, or certain types of cryptographic challenges that are currently considered secure against classical computational attacks. The achievement of quantum advantage suggests that for these select problems, quantum computing can offer transformative solutions, unlocking new possibilities in fields ranging from materials science to cryptography.

It's important to distinguish between quantum advantage and quantum supremacy. Quantum supremacy is achieved when a quantum computer can solve a problem that is practically impossible for classical computers to solve within a reasonable timeframe, regardless of whether the problem has practical application. Quantum advantage, on the other hand, stresses the practical utility of quantum computing in solving real-world problems more efficiently than classical systems, marking a pivotal step toward the widespread adoption and development of quantum technology.

In this paper, we have tackled the promise of Quantum Advantage, for a specific problem, the calculation of Solvency Capital Requirement for Internal Model firms in Europe. The theoretical foundation was laid out by Egger et al [2], where they proved that in theory a near quadratic speedup is possible. However, their analysis did not consider in any detail the differences in the clock speeds of classical and quantum computers, just that a quantum computer could calculate VaR in fewer steps.

In this section we examine this claim in a bit more detail.

7.1. Why Polynomials?

Stefan Woerner and Daniel J. Egger, in their work on quantum risk analysis [2], used polynomials to approximate the Profit and Loss (PnL) of a portfolio for several reasons, which are closely tied to the inherent strengths and operational paradigms of quantum computing as well as the specific needs of financial risk analysis:

1. **Quantum Amplitude Estimation (QAE):** Quantum computing offers unique advantages in certain computational tasks, among which is Quantum Amplitude Estimation (QAE). QAE can provide quadratic speedup for estimating the mean of a quantum observable. However, QAE requires the input to be encoded in the amplitudes of a quantum state. By approximating the PnL distribution of a portfolio with polynomials, it becomes feasible to efficiently encode this information into quantum states, leveraging the quantum speedup offered by QAE for risk analysis tasks like Value at Risk (VaR) and Conditional Value at Risk (CVaR).
2. **Simplicity and Efficiency:** Polynomial functions are mathematically simple and computationally efficient to evaluate, both on classical and quantum computers. This simplicity aids in the practical implementation of the approximation within the circuit design of a quantum algorithm. Moreover, the efficiency of polynomial evaluations means that the resource requirements (in terms of qubits and quantum gates) are kept within feasible limits, making the approach more accessible with current and near-term quantum technology.
3. **Flexibility in Approximation:** Polynomials can approximate a wide range of functions to a desired degree of accuracy by adjusting the degree of the polynomial. This flexibility is crucial in financial applications where the PnL distribution can take various forms depending on the underlying assets and market conditions. By using polynomials, Woerner and Egger ensured that their methodology could be adapted to different portfolios and risk scenarios without requiring a complete redesign of the quantum algorithm.
4. **Compatibility with Quantum Algorithms:** Quantum computers are inherently probabilistic, and many quantum algorithms naturally produce outcomes in

terms of probability amplitudes. The polynomial approximation of PnL fits well into this paradigm, as it allows for a direct mapping between the financial metrics being estimated and the probabilistic outcomes of quantum measurements.

5. **Analytical Differentiability:** Polynomial functions are analytically differentiable, which is a valuable property for optimization and sensitivity analysis. In the context of quantum risk analysis, this allows for the direct calculation of risk sensitivities and gradients, which are essential for portfolio optimization and risk management.

The use of polynomials in both quantum risk analysis and the Solvency Capital Requirement (SCR) calculations by insurance companies with Internal Models is not entirely coincidental. Both applications leverage polynomials for their beneficial mathematical and computational properties, although the underlying motivations and contexts differ. Here are a few reasons why polynomials find utility in these seemingly disparate areas:

Common Mathematical Advantages

- **Flexibility in Function Approximation:** Polynomials can approximate a wide range of functions accurately, making them a versatile tool for modelling complex financial metrics, including those related to risk and return.
- **Analytical Properties:** Polynomials are easy to differentiate and integrate, which is advantageous in optimization problems, sensitivity analysis, and in calculating gradients required for risk management and regulatory compliance.
- **Computational Simplicity:** Polynomial functions are computationally straightforward to evaluate, which can lead to efficient algorithms that are easier to implement and run on both classical and quantum computers.

SCR Calculations in Insurance

For insurance companies, the Solvency Capital Requirement (SCR) is a critical regulatory metric under the Solvency II framework, which aims to ensure that insurers hold enough capital to cover their risks. Internal Models used by insurers for SCR calculations are designed to accurately reflect the unique risk profile of their liabilities and assets. Polynomials can be employed within these models to approximate complex relationships and dynamics in insurance risks, including underwriting risk, market risk, credit risk, and operational risk.

Convergence on Efficient Methods

- **Quantitative Finance and Risk Management:** Both quantum risk analysis and SCR calculations are deeply rooted in quantitative finance and risk management. As such, they draw from a common pool of mathematical and statistical tools designed to model financial risks and returns.
- **Optimization Under Constraints:** Insurance companies and financial institutions often face the challenge of optimizing their portfolios under various constraints, including regulatory requirements. Polynomials offer a manageable way to encode these constraints and objectives into mathematical models that can be optimized.

Divergent Contexts, Similar Tools

While the use of polynomials in both contexts arises from their mathematical convenience and computational efficiency, the ultimate goals are somewhat different. In quantum computing, the motivation includes leveraging the unique capabilities of quantum algorithms for speedup and efficiency gains. In the context of SCR calculations, the focus is more on timely calculation of capital requirements for risk management and regulatory disclosures.

7.2. Clock Speeds and Coherence times.

In this section we briefly explore the difficulties associated with comparing the clock speeds of classical computers, such as those with an Intel i9 processor, to quantum computers, like those used in the IBM Condor processor. One realises we are looking at fundamentally different technologies with distinct operational principles, and a direct comparison is misguided. Let's outline the differences in the context of clock speeds and operational paradigms.

Classical Computers (e.g., Intel i9)

- **Clock Speed:** The clock speed of a classical processor, like the Intel i9, is a measure of its operational frequency and is typically represented in gigahertz (GHz). This means billions of cycles per second. For instance, an Intel Core i9 processor might have a base clock speed ranging from 3.6 GHz to 5.3 GHz, depending on the specific model and generation. This speed indicates how many instruction cycles the processor can perform in a second.
- **Operational Principle:** Classical computers operate on binary information, processing bits that are either 0s or 1s. The clock speed directly influences how fast these bits can be processed, with higher speeds indicating faster processing capabilities.

Quantum Computers (e.g., IBM Condor)

- **Clock Speed:** The concept of "clock speed" as it applies to classical computers doesn't directly translate to quantum computers. Quantum computers operate on qubits, which can represent 0, 1, or any quantum superposition of these states. The performance of a quantum computer is not dictated by clock speed but by the coherence time, error rates, and the number of qubits. The IBM Condor processor, aiming to be one of the most advanced quantum processors with a significant number of qubits, doesn't have a "clock speed" in the traditional sense.
- **Operational Principle:** Quantum computers leverage the principles of quantum mechanics, including superposition, entanglement, and interference, to perform calculations. The efficiency and capability of a quantum computer are more about its ability to maintain the quantum state of its qubits (coherence time), the quality of qubit interactions (gate fidelities), and its capacity to scale up (number of qubits) without significant increases in error rates.

Key Differences

- **Measurement of Performance:** Classical computer performance is often gauged by clock speed, among other factors like core count and cache size. In contrast, quantum computer performance is measured by coherence time, error rates, qubit count, and the ability to perform error correction.
- **Operational Paradigm:** Classical processors process information in a binary format, executing instructions one after another or in parallel across multiple cores. Quantum processors perform operations on qubits that can exist in multiple states simultaneously, allowing them to explore a vast computational space with fewer operations but under constraints of quantum coherence and error rates.

7.3. Quantum Disadvantage before Quantum Advantage

What disadvantages stem from trying to solve this problem using a quantum computer?

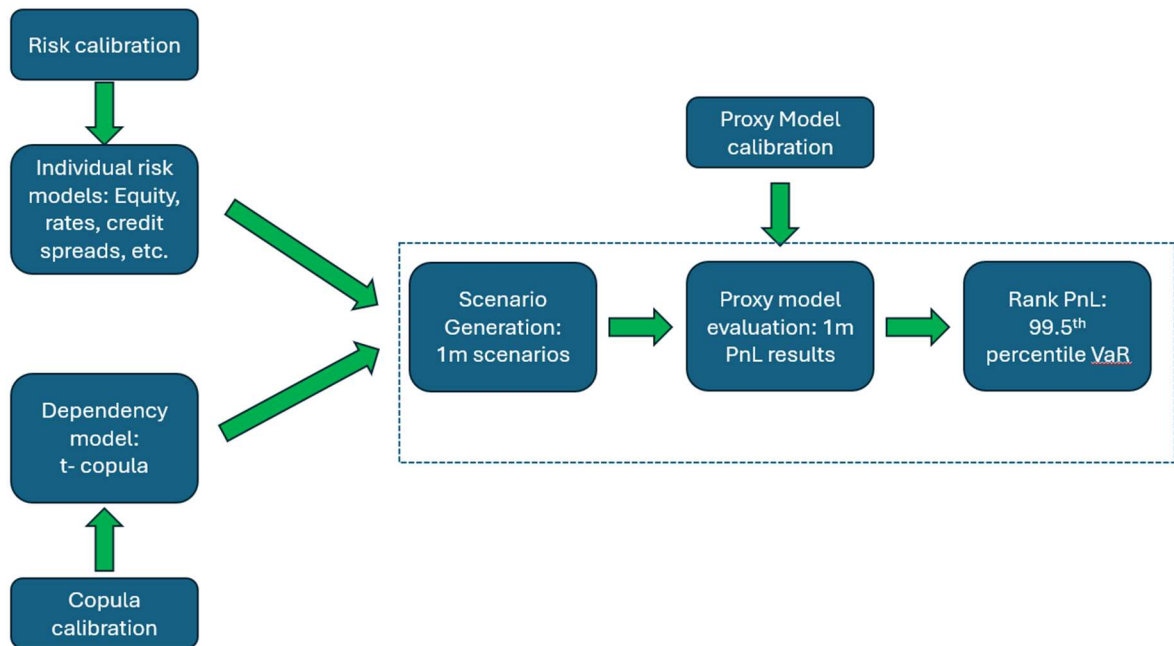
Whilst the approach set out by Egger et al. [2] promises to calculate VaR in fewer steps (and simulations) compared to its classical counterpart, this does not tell us whether calculating SCR using a quantum computer will deliver any quantum advantage.

As set out in the section above, a direct comparison between the clock speeds is potentially misguided, so how do we assess if Quantum Computing confers an advantage?

A simple way to answer this question is to consider whether one can speed up the calculation by replacing classical computers with quantum computers for specific parts or the whole calculation.

Let's briefly recap the stages of an SCR calculation for an Internal Model firm.

Figure 15: End to end SCR calculation.

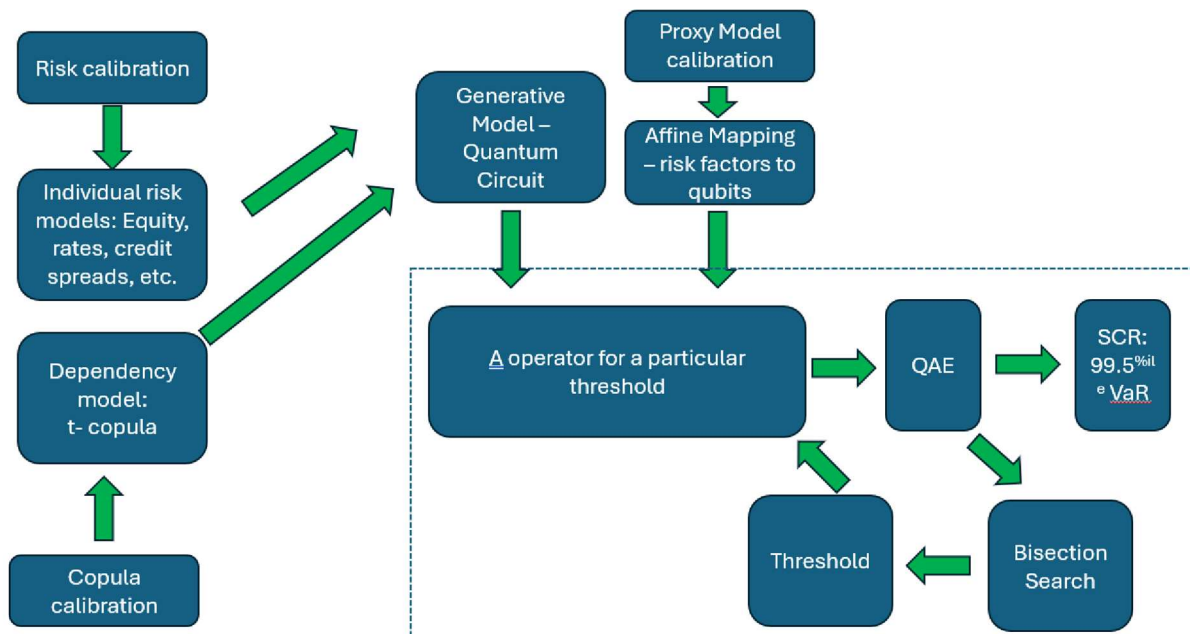


The figure above shows the processes involved in the end-to-end calculation of the SCR. The dotted box shows the components of the calculation that we have identified for quantum acceleration.

However, the above schematic is not representative of the end-to-end calculation when utilising a quantum computer, as it will entail additional steps for making the dotted box 'quantum ready'.

The figure below illustrates the process with quantum acceleration.

Figure 16: SCR calculation with quantum acceleration.



A comparative analysis of **Figure 15** and **Figure 16** reveals that the quantum-accelerated approach involves a greater number of procedural steps than its classical counterpart. This observation is pivotal when assessing the practical benefits of quantum advantage, necessitating a careful evaluation of the complexities introduced by these additional steps in relation to the overall speedup achieved.

The complexity primarily emanates from two critical components:

- 1. Generative Model:**

The calibration of individual risks and the copula within a quantum framework mandates intricate processes. These can be approached through:

- a) Exact Implementation:** Necessitating a circuit depth that grows exponentially with the number of qubits assigned to represent risk factors. For instance, with 31 risk factors, each represented by 20 qubits, the circuit depth would approximate $O(2^{620})$, rendering it impractically deep for execution within

reasonable timeframes. Moreover, Monte-Carlo sampling fails to provide sufficient basis state diversity for accurate circuit initialization.

b) **Quantum Machine Learning (QML):** We introduced a QML strategy known as QCBM for encoding probability distributions into qubits at a predetermined circuit depth, extendable to multivariate distributions. Nevertheless, when dealing with numerous risk factors and their interdependencies, the model's 'training' phase might extend significantly, marking a notable consideration in terms of overhead.

c) **Quantum Analogues to Parametric Distributions and Copulas:**

Traditionally, parametric distributions serve as efficient approximations for modelling underlying processes or risks at an acceptable computational expense. Although these models never perfectly match the data, they are invaluable for interpolation and extrapolation in the absence of direct observational data. In the quantum context, one could streamline the process by transitioning directly from data to quantum-efficient distributions, bypassing traditional parametric fitting. This suggests a more direct method of leveraging quantum computing for generative risk modelling, potentially reducing complexity and enhancing efficiency.

2. Proxy Model Mapping:

In [Section 6.2.1](#), it is outlined that proxy models typically function by mapping the values of risk variables directly to net asset value (NAV) or profit and loss (PnL). However, a direct application of these models is not feasible due to the necessity of translating these mappings into a form compatible with quantum bits (qubits). This translation process introduces significant overhead.

a) Affine Mapping from Risk Factors to Qubits:

Affine mappings are essential for converting polynomial expressions of risk factors into equivalent qubit representations. For instance, a risk factor x represented by three qubits, $|q_2q_1q_0\rangle$, in a range $[0,7]$, would be transformed into $x=q_0+2q_1+4q_2$ via affine mapping. This transformation process, however, is notably complex, with different strategies presenting their own advantages and challenges:

- **Computational Graph Construction:** While this method allows for systematic substitutions within the polynomial, it is highly memory intensive. Even at moderate problem scales, this approach can lead to significant memory constraints.
- **Numerical Substitution:** This alternative, which involves the use of linear algebra techniques for substitution, offers a balance between memory usage

and computational efficiency. Unfortunately, due to the size of the matrices involved, this method may still necessitate some degree of iterative computation, significantly slowing down the process compared to direct model evaluations.

b) Implementing the proxy model on a quantum computer:

Section 6.2.1 illustrated the mapping from risk factor polynomial to a qubit polynomial, reproduced below:

Risk Factor Polynomial					
# Term	RiskFactor1	RiskFactor2	Power1	Power2	Coefficient
1	x	None	1	None	2
2	y	None	2	None	3
3	x	y	1	1	-1

Qubit polynomial					
# Term	RiskFactor1	RiskFactor2	Power1	Power2	Coefficient
1	q0	None	1	None	2
2	q1	None	1	None	4
3	q2	None	1	None	3
4	q3	None	1	None	12
5	q0	q2	1	1	-1
6	q0	q3	1	1	-2
7	q1	q2	1	1	-2
8	q1	q3	1	1	-4
9	q2	q3	1	1	12

Whilst terms 1-4 in the qubit representation are straightforward, terms 5-9 can be constructed via multi-controlled X gates targeted at qubits initialised to $|0\rangle$, these will be the interaction qubits. The coefficients for each of these terms can be captured via the 'Weighted Addition' procedure set out in [Section 6.2.2](#).

As shown above, the transition from a risk factor polynomial to a qubit-based polynomial introduces a substantial increase in complexity. For example, a proxy model with 863 terms can expand into a qubit polynomial with more than 35 million terms. While the initial terms of such a qubit polynomial might be straightforward to construct, the complexity escalates with terms representing interactions, which require multi-controlled X gates and careful initialization of interaction qubits.

The implementation of such extensive quantum circuits is exceedingly time-consuming. Constructing a quantum circuit for a relatively simpler model with around 50,000 terms, as opposed to 35 million, can take upwards of 16 hours. This significant time investment highlights the practical challenges in translating complex proxy models into quantum computational frameworks.

7.4. But which is Faster?

In the realm of financial engineering, the determination of whether quantum computing offers a tangible advantage over classical computing is intricately tied to the specific timeframe considered for comparison. This nuance is well understood by seasoned quants and financial engineers. For instance, after the construction phase of a quantum circuit is complete, running a Solvency Capital Requirement (SCR) calculation on a quantum computer might indeed prove faster than performing the same task on a classical system. However, this perspective overlooks substantial preliminary overheads, casting doubt on the practical feasibility of achieving a genuine 'quantum advantage' in this context. The time required to design and build the quantum circuit is substantial, during which millions of SCR calculations could potentially be executed on a classical computer.

The performance benchmark for classical computing, which serves as the reference point for these comparisons, is influenced by a multitude of factors:

1. **Model Complexity:** The efficiency of computation on classical computers varies significantly with the complexity of the model in question. Factors such as the size and order of the polynomial, the number of cross terms, and the total number of risk factors involved play a crucial role.
2. **Hardware Capabilities:** The type of hardware used can greatly impact computation speed. Graphics Processing Units (GPUs) and Field-Programmable Gate Arrays (FPGAs) often outperform Central Processing Units (CPUs) due to their superior processing speed, memory capacity, and ability to handle parallel computations.
3. **Programming Language:** The choice of programming language can also affect performance. Lower-level languages such as C++ are typically faster and more efficient than higher-level languages like Python or Julia, especially for computationally intensive tasks.
4. **Algorithmic Efficiency:** The specific numerical methods employed for simulation are critical. Ongoing advancements in algorithms can drastically

reduce computation times, enhancing the overall efficiency of classical computing solutions.

Understanding these factors is essential for accurately assessing the potential for quantum computing to offer advantages in financial applications. It underscores the importance of considering not just the raw computational speed post-circuit construction but also the preparatory stages and inherent limitations that may impact the overall viability of quantum computing for tasks such as SCR calculations. The dialogue between quantum and classical computing capabilities continues to evolve, with each of these considerations playing a pivotal role in shaping the future of financial engineering and quantitative analysis.

7.5. What about Noisy Qubits?

In the discussion so far, we have only considered 'logical qubits', which is a term used for 'idealised' qubits that have very low error rates and quantum gates that have a very high fidelity.

However, we are currently in the Noisy Intermediate State Quantum (NISQ) era, which means that the quantum computers available to us are noisy and have limited gate fidelity. This means that 'useful' computation requires a level of redundancy within the quantum system for error correction.

Quantum Error Correction (QEC) schemes, including widely studied protocols like the surface code, are designed to overcome limitations in gate fidelity and other types of errors (such as qubit decoherence and initialization errors) that quantum computers face. Gate fidelity refers to the accuracy with which quantum gates—operations that change the state of qubits—can be performed. High-fidelity gates are crucial for quantum computing because errors in gate operations can propagate and undermine the computation's correctness. Here's how QEC schemes address the challenge of limited gate fidelity:

1. Encoding Logical Qubits in Multiple Physical Qubits

QEC schemes work by encoding the information of one logical qubit into a highly entangled state spread across multiple physical qubits. This redundancy allows the system to detect and correct errors without measuring the quantum information directly, which would otherwise collapse the quantum state due to the measurement postulate of quantum mechanics.

2. Syndrome Measurement

Error correction codes use syndrome measurements to identify errors. These measurements involve ancilla (helper) qubits that are entangled with the logical qubits' physical qubits. By measuring the ancilla qubits, the system can infer the

presence and type of errors on the logical qubits without disturbing their quantum state. This process allows for the detection of both bit-flip and phase-flip errors, which are common types of errors in quantum computing.

3. Fault-Tolerant Gate Operations

To address gate fidelity directly, QEC protocols implement fault-tolerant gate operations. This means that even if a gate operation introduces an error, the error can be detected and corrected without causing a failure in the computation. Fault-tolerant gate constructions often involve sequences of physical gate operations that, collectively, perform a logical gate operation on the logical qubits. These sequences are designed so that any single gate error can be corrected by the error correction protocol.

4. Threshold Theorem

The threshold theorem for fault-tolerant quantum computing states that if the error rate per gate and per qubit is below a certain threshold, an arbitrary long quantum computation can be performed reliably by sufficiently enlarging the error-correcting code. This theorem underpins the entire field of quantum error correction and is the reason why developing high-fidelity gates and error correction codes is so critical. If gate fidelities can be improved beyond this threshold, quantum error correction schemes can effectively suppress errors, allowing for scalable and reliable quantum computations.

5. Error Correction and Gate Improvement Cycles

In practice, error correction schemes and improvements in gate fidelity go hand in hand. As gate fidelities improve, the requirements for error correction (in terms of the number of physical qubits needed per logical qubit) can become less stringent, making quantum computations more resource efficient. Conversely, effective error correction schemes can make certain computations feasible even with current gate fidelity levels.

Quantum error correction schemes address the challenge of limited gate fidelity by spreading information across multiple qubits, employing syndrome measurements for error detection, implementing fault-tolerant operations, and relying on the threshold theorem to enable scalable quantum computing.

One of the most efficient and widely studied quantum error correction protocols is the **surface code**. The efficiency of an error correction protocol in quantum computing is typically measured by its ability to correct errors with minimal overhead in terms of physical qubits required for each logical qubit, while also being compatible with the operational constraints of the quantum computing architecture, such as nearest-neighbour interactions.

Surface Code

- **Efficiency:** The surface code is highly regarded for its relatively high threshold error rate (approximately 1% under certain conditions), meaning that if the physical error rates of qubits and operations can be kept below this threshold, the surface code can effectively correct errors. This makes it one of the most promising protocols for large-scale fault-tolerant quantum computing.
- **Qubit Overhead:** The exact overhead in qubit requirements depends on the desired error rate of the logical qubits (how often a logical qubit might experience an error after correction). To achieve fault tolerance, the surface code typically requires a grid of qubits, with a significant overhead: each logical qubit might require hundreds to thousands of physical qubits, depending on the quality (error rate) of the physical qubits and the level of error correction desired.

Considerations

- **Physical vs. Logical Qubits:** It's important to differentiate between physical qubits, which are the actual quantum bits in a quantum computer, and logical qubits, which are error-corrected qubits formed from multiple physical qubits using quantum error correction codes. The overhead comes from the need to use many physical qubits to create a single, more reliable logical qubit.
- **Threshold Error Rate:** The threshold error rate is a critical parameter that defines the maximum physical error rate that can be tolerated while still successfully correcting errors. Surpassing this rate makes fault-tolerant quantum computing infeasible, and protocols like the surface code are designed to maximize this threshold.

While **surface code** is notable for its practicality and threshold, other protocols like the **toric code** (closely related to the surface code) and **colour codes** also offer paths to error correction, each with its trade-offs in terms of error tolerance, qubit overhead, and operational complexity.

Each method significantly increases the qubit requirement for quantum computing, often necessitating an order of magnitude more physical qubits than logical qubits to ensure fault tolerance.

The Case-Study explored in this paper requires c36m logical qubits, which means that roughly 36bn physical qubits will be required to perform useful computation using today's noisy qubits, and known error correction schemes. This is a very large qubit requirement for a problem that can be solved on a personal laptop within seconds.

As quantum computing technology evolves, the efficiency of error correction protocols and the reduction in physical qubit requirements for effective error correction will be crucial areas of development.

7.6. The Quantum Conundrum: Is it worth it?

The sections above have outlined many challenges associated with using quantum computers for the practical application of calculating an SCR for a medium sized firm.

This is an application for which classical computing is already very efficient, even using personal laptops, and modestly efficient code. So why bother with quantum computing?

We will answer this question in two parts, part one considers the application itself, part 2 more general.

Quantum Risk Appetite

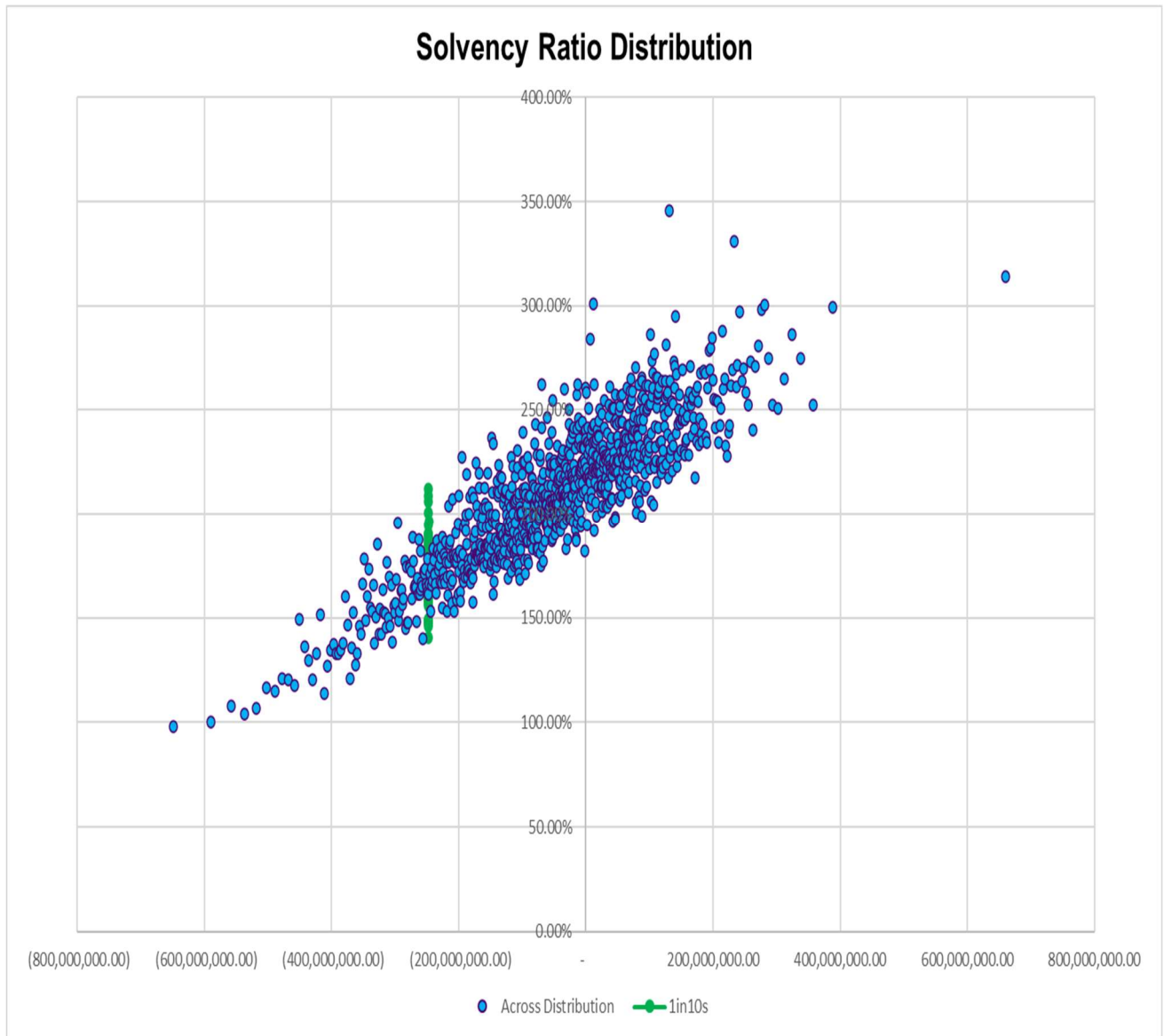
While it is true that Solvency Capital Requirements (SCRs) can be effectively computed with classical computing methods, the sole purpose of an Internal Model is not just to calculate the SCR. Under Solvency II, companies must integrate their Internal Models into their decision-making processes, proving they meet the 'Use-Test' criteria.

Companies often define their risk tolerances and constraints according to the calibrations of their Internal Model. Commonly, companies use a singular scenario that represents the 1-in-10-year events for specific risks, and a single multivariate loss scenario.

Though this approach may be more beneficial than taking no action, and it allows companies to claim they've utilized the Internal Model in setting risk appetites and limits, it doesn't fully leverage the potential offered by an internal model. Particularly, as the author observes, the 1-in-10-year scenarios that companies typically focus on for

setting their risk appetites and limits vary widely in their impact on the Solvency Ratio, as depicted in the figure below.

Figure 17: Solvency Ratio distribution.



The chart displays the Solvency Ratio distribution for a hypothetical company, with the Solvency Ratio calculated as follows:

$$SR = NAV / SCR$$

where NAV represents the Net Asset Value (Own Funds), and SCR stands for the Solvency Capital Requirement.

The chart illustrates the process of calculating the SCR, which involves simulating the Net Asset Value of the firm across numerous scenarios and determining the 99.5th

percentile of the worst-case variation in NAV. However, companies are not required to estimate what the SCR would be in those specific scenarios.

During the process of setting risk appetites and limits, firms explore what the world might resemble after a fairly moderate 1-in-10-year adverse event, considering its influence on the SCR and Solvency Ratio. The plot captures both the change in NAV on the horizontal axis and the Solvency Ratio after the SCR recalculation on the vertical axis. The blue dots depict scenarios from the complete distribution of NAV, while the green dots represent the 1-in-10-year event scenarios. Observers will note that even though the green dots signify the same NAV loss (indicated by their horizontal axis position), they result in quite disparate SCR outcomes (vertical axis position). Notably, it demonstrates the sensitivity of the Solvency Ratio, even in moderately adverse scenarios. This sensitivity is often not accounted for in the processes of setting risk appetite and limits, potentially leading to less-than-optimal decisions for risk mitigation and portfolio management.

Regarding quantum computing, although it might not justify the overhead costs to compute the SCR once, the equation changes when there is a need to calculate the SCR numerous times to gain a comprehensive understanding of the risk landscape. It's also pertinent to mention that the risk appetite process usually happens outside of the regular business cycle in many companies, meaning that the additional time required to reformulate the problem for quantum processing shouldn't introduce delays for this purpose.

The General Case

Quantum computing stands at the forefront of the next technological revolution, offering the potential to transform industries, solve previously intractable problems, and redefine computational boundaries. While the journey to realize its full potential is fraught with technical, conceptual, and operational challenges, the pursuit of quantum computing is justified by several compelling reasons:

1. **Unprecedented Computational Power:** Quantum computers leverage the principles of quantum mechanics to process information in ways fundamentally different from classical computers. By exploiting superposition and entanglement, quantum machines can process vast amounts of data simultaneously, offering exponential speedups for certain calculations.
2. **Solving Complex Problems:** Quantum computing has the potential to address problems that are currently beyond the reach of classical computers. This includes simulating complex molecular interactions for drug discovery,

optimizing large-scale logistics and supply chains, and providing new insights into natural phenomena through simulation.

3. **Advancing Cryptography:** Quantum computers could revolutionize the field of cryptography. While they pose a threat to current encryption methods, they also pave the way for quantum encryption technologies, like quantum key distribution, which could provide levels of security that are theoretically unbreakable under the laws of quantum physics.

4. **Accelerating Machine Learning:** Quantum algorithms could dramatically accelerate machine learning processes, making it possible to analyse larger datasets and perform more complex pattern recognition than is currently achievable, potentially leading to breakthroughs in artificial intelligence.

5. **Economic and Strategic Advantage:** The nations and organizations that master quantum computing technology are poised to gain significant economic and strategic advantages. This could impact national security, economic competitiveness, and technological leadership, motivating sustained investment and research despite challenges.

6. **Boosting Scientific Research:** Quantum computing offers powerful tools for scientists across disciplines. It could lead to new discoveries in physics, chemistry, biology, and beyond, as it allows for the simulation and analysis of phenomena that are too complex for classical computation.

7. **Enabling Scientific Breakthroughs:** Quantum computing holds the key to scientific advancements in fundamental physics, including the mysteries of dark matter and the intricacies of quantum gravity, potentially leading to a deeper understanding of the universe and our place within it.

8. **Long-term Technological Progress:** Like the space race of the 20th century, the pursuit of quantum computing drives progress in numerous ancillary fields, including materials science, engineering, and software development, furthering overall technological advancement.

9. **Inspiring a New Generation of Scientists and Engineers:** The challenges of quantum computing are inspiring a new generation of problem solvers. Educating and training this talent pool not only prepares us for the quantum era but also enriches the broader scientific community.

10. **Sustainable Solutions:** Quantum computing could optimize energy consumption, waste reduction, and resource allocation, contributing to more sustainable solutions and helping tackle climate change.

In conclusion, while quantum computing presents significant challenges, the potential rewards justify the global effort in overcoming these obstacles. The investment of time,

resources, and intellect in quantum computing is not just a pursuit of computational power; it is an investment in a future that promises to be fundamentally richer in possibilities and innovations across all facets of human endeavour.

7.6. Avenues for future research on this topic.

The preceding sections have critically examined the formidable challenges associated with harnessing quantum advantage for Value at Risk (VaR) calculations. Some challenges are the subject of active research, such as the advancement of Error Correction Schemes and the pursuit of the optimal qubit implementation paradigm. However, alongside these complex avenues, there exist promising research paths that may offer more immediate progress. Section 7.3 delved into the specific bottlenecks encountered when translating the VaR calculation into the quantum domain.

1. **Efficient Encoding of Multivariate Distributions:** Our discussion highlighted innovative approaches like Quantum Circuit Born Machines (QCBMs) and introduced Quantum Generative Adversarial Networks (QGANs) as alternative methods. These quantum algorithms have the potential to streamline qubit requirements and circuit depths, yet they come with the significant challenge of training these models. Future research should prioritize the discovery of 'quantum analogues of parametric distributions'—quantum-native methods that can represent complex statistical distributions with quantum efficiency.
2. **Efficient Mapping of Polynomials to Qubits:** To leverage a quantum computer for SCR calculations, one must translate the proxy model—used to represent a company's Net Asset Value (NAV) or Profit and Loss (PnL)—into a quantum-compatible format. This conversion necessitates the construction of an affine map that relates each risk factor to its corresponding qubits and then reconstitutes the proxy model by replacing risk factors with their quantum representations. Advancements in algorithmic design are necessary to make this process more streamlined and less resource-intensive.
3. **Efficient Construction of Large Quantum Circuits:** From the author's practical experience, the most compute-intensive aspect of the calculation was assembling the quantum circuit for the A operator, as dictated by the canonical Quantum Amplitude Estimation (QAE) method for the chosen proxy model. There is a pressing need for the developers of quantum software frameworks, like Qiskit, to innovate on this front. Enhancing the tools and algorithms for quantum circuit construction could significantly reduce computational overhead and time investment.

In each of these three areas, there is considerable scope for improvement. By developing more sophisticated quantum algorithms and software tools, we can mitigate the current limitations in the application of quantum computing to financial modelling. This progress is essential not only for achieving practical quantum advantage in financial risk assessments but also for pushing the boundaries of what is computationally achievable, laying the groundwork for future quantum-enabled discoveries across various disciplines.

8. Conclusions

This paper critically assesses the potential for 'quantum advantage' in risk management applications, such as the calculation of Value at Risk (VaR) using quantum computers for practical problems. It is clear that the journey towards this goal is filled with complexities, and that significant hurdles will need to be overcome. However, the evolving landscape of quantum computing shows promise of overcoming these challenges over time.

Classical Realm Considerations for Quantum Readiness:

1. **Quantum-Compatible Problem Formulation:** Transitioning the VaR calculation to a quantum-ready format is non-trivial, and potentially more expensive computationally than calculating VaR using a classical computer in the first place. Further research can help to discover algorithms to streamline this process.
2. **Efficient Quantum Circuit Design:** The development of large quantum circuits is currently a significant hurdle, yet the progress in quantum algorithms and software tools is encouraging. It is plausible that more intuitive and efficient methods for circuit design will emerge as the field matures.

Quantum Realm Progress and Pragmatism:

1. **Qubit Quantity and Quality:** While today's quantum computers would require an impractically high number of qubits to match the performance of a personal laptop in certain tasks (36 billion physical qubits for the task examined in this paper), ongoing advancements in quantum error correction and qubit coherence are anticipated to lower these requirements gradually and sustainably.
2. **Advancements in Quantum Hardware:** The size and fidelity of quantum computers impose significant constraints on the types of problems that can be tackled with present day quantum computers. However, considering the rapid pace of technological development, more refined and less noisy quantum systems are expected to become available. This evolution will likely make

quantum applications more viable for specific use cases, including those in risk management.

In summary, the path towards realizing quantum advantage in practical financial applications is nuanced and requires a balanced view. While we are witnessing considerable progress, the full potential of quantum computing in this context remains on the horizon. This paper presents a realistic appraisal of the current state of quantum computing, acknowledging both its limitations and its potential, with a grounded expectation that continued research and development will eventually enable quantum computers to meaningfully contribute to risk management and other complex financial calculations.

References

1. Preskill, J. (2018). Quantum Computing in the NISQ era and beyond. *Quantum*, **2**, 79.
2. Egger, D. J., Gutiérrez, R. M., Mestre, J. C., & Woerner, S. (2019). Quantum risk management. *Quantum Finance*, **1**, 105-112.
3. Egger, D. J., Gutiérrez, R. M., Mestre, J. C., & Woerner, S. (2020). Quantum computing for finance: Overview and prospects. *IEEE Transactions on Quantum Engineering*, **1**, 3100915.
4. Rudin, W. (1976). *Principles of Mathematical Analysis* (3rd ed.). McGraw-Hill.
5. Jacquier, A., Kondratyev, O., Lipton, A., & López de Prado, M. (2022). *Quantum Machine Learning and Optimisation in Finance*. Packt Publishing.
6. Cuccaro, S. A., Draper, T. G., Kutin, S. A., & Moulton, D. P. (2004). [A new quantum ripple-carry addition circuit](#)*arXiv:quant-ph/0410184v1*
7. Brassard, G., Hoyer, P., Mosca, M. & Tapp, A. Quantum amplitude amplification and estimation. *Contemporary Mathematics* 305, <https://doi.org/10.1090/conm/305> (2002).
8. Iterative Quantum Amplitude Estimation. Grinko, D., Gacon, J., Zoufal, C., & Woerner, S. (2019). <https://arxiv.org/abs/1912.05559>
9. Amplitude Estimation without Phase Estimation. Suzuki, Y., Uno, S., Raymond, R., Tanaka, T., Onodera, T., & Yamamoto, N. (2019). <https://arxiv.org/abs/1904.10246>
10. Faster Amplitude Estimation. K. Nakaji (2020). <https://arxiv.org/pdf/2003.02417.pdf>