British Actuarial Journal (2025), Vol. **30**, e1, pp. 1–41
doi:10.1017/S135732172400031X

CONTRIBUTED PAPER

Quantum internal models for Solvency II and quantitative risk management

Muhammad Ahmer Amjad

Email: ahmeramjad@yahoo.com

Abstract
This paper extends previous research on using quantum computers for risk management 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, we closely examine the practical bottlenecks in developing and maintaining quantum internal models. Our work seeks to determine whether a quadratic speedup, through quantum amplitude estimation can be realised for problems at an industrial scale. It also builds on previous work that explores the application of quantum computing to the problem of asset liability management in an actuarial context. Finally, we identify both the obstacles and the potential opportunities that emerge from applying quantum computing to the field of insurance risk management.

Keywords: Quantum computing; risk management; Solvency II; insurance

1. 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 characterised 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).

Building on their previous research, Egger et al. (2020) illustrated that quantum computers can achieve a quadratic speedup over classical counterparts for Value at Risk(VaR) of a credit portfolio. Quadratic speedup is a term from computer science that refers to a situation where a problem can be solved in $\sqrt{N}$ steps as opposed to N steps by using a quantum algorithm instead of the best classical algorithm. These investigations, however, were limited to stylised 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 readers. Then, we assess whether a quantum advantage exists for a VaR calculation for a sufficiently large and complex portfolio. In particular, we try to answer the following questions:

(1) How is a quantum internal model different from the types of problems discussed in previous research?

© The Author(s), 2025. Published by Cambridge University Press on behalf of The Institute and Faculty of Actuaries. This is an Open Access article, distributed under the terms of the Creative Commons Attribution licence (https://creativecommons.org/licenses/by/4.0/), which permits unrestricted re-use, distribution, and reproduction in any medium, provided the original work is properly cited.

(2) How can the methods used for solving stylised problems in previous papers be extended for building a quantum internal model?
(3) Does a quantum advantage exist for such problems?
(4) Is there scope for a quantum advantage in other, similar problems?

2. Internal models for Solvency II firms

2.1 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 (Pillar1), 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 requires 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.

2.2 Internal models

The primary function of an Internal Model is to compute the SCR, which employs a 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 (McNeil et al., 2015):

1. **Individual risk modules**: These modules typically utilise parametric distributions to model the behaviour of singular risk factors. For instance, the Lognormal distribution might be employed to simulate equity price returns.
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 (NAV), 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 generates hundreds of thousands of simulations of potential outcomes over the forthcoming year.

The asset and liability valuation module then translates these scenarios into NAV impacts (NAV or profit and loss PL), 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 approximate the valuation of any asset or liability, or even the 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 (Rudin, 1976). 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 Section 3.

3. 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 realisations of quantum computers differ among providers, it is essential to explore the distinctions between quantum and classical computing from a conceptual 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, specifiable with longitude and latitude coordinates.

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 NISQ era of quantum computing is characterised 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.

3.1. Gate based Model of Quantum Computation

The gate-based model of quantum computation (Nielsen and Chuang, 2010) is the most widely recognised 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:

- **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 analyser or a slide rule).

3.1.1. 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.

Specialised problem-solving: It's specifically designed to solve optimisation 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. In quantum mechanics the Hamiltonian is an operator corresponding to the total energy of the system, including kinetic and potential energies. For quantum computing the Hamiltonian governs the time evolution of the qubits. For example, quantum computers like those developed by D-Wave Systems use quantum annealing based on Hamiltonian dynamics.

Not universal: While powerful for optimisation, quantum annealing isn't known to be universal for quantum computation and is tailored to specific types of problems rather than general purpose computation.

3.1.2. 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 specialised for optimisation 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 optimisation problem.

Hardware implementation: Quantum annealing is typically implemented in quantum annealers like D-Wave systems, which are designed to solve optimisation problems. In contrast, gate-based quantum computers require a set of universal quantum gates and typically aim to be more general-purpose.

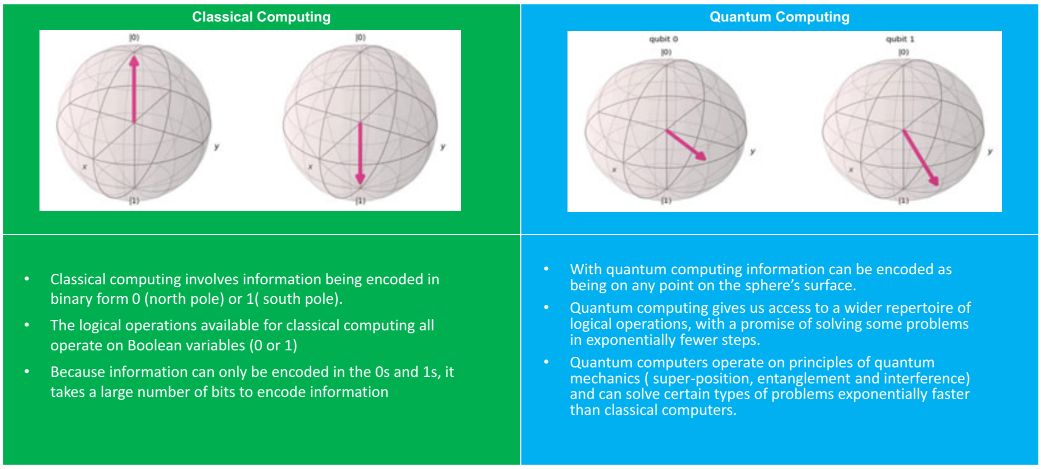
Error correction: Gate-based models have developed error correction protocols to deal with the fragility of quantum states, which is crucial for their scalability. Quantum annealing also deals with noise and errors but in a different manner, often leveraging the robustness of the ground state search 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 specialised and does not provide a universal computational model.

3.2. Quantum versus 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 (Figure 1 below). 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.



- Classical computing involves information being encoded in binary form 0 (north pole) or 1(south pole).
- The logical operations available for classical computing all operate on Boolean variables (0 or 1)
- Because information can only be encoded in the 0s and 1s, it takes a large number of bits to encode information

- With quantum computing information can be encoded as being on any point on the sphere's surface.
- Quantum computing gives us access to a wider repertoire of logical operations, with a promise of solving some problems in exponentially fewer steps.
- Quantum computers operate on principles of quantum mechanics (super-position, entanglement and interference) and can solve certain types of problems exponentially faster than classical computers.

Figure 1. Difference between classical bits and quantum bits.

3.3. Quantum versus 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 and 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 can perform all operations that classical gates do and more, but they are 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 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 or quadratic speedups in some cases (Tables 1 and 2).

Table 1. Types of quantum gates

Quantum gates (Examples)	Description
Hadamard (H)	Creates a superposition of |0> and |1>
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 probabilities of finding the qubit in |0> and |1> unchanged.
CNOT	Conditional NOT, flips the target qubit only if the control qubit is in the state |1>.
Toffoli (CCNOT)	Acts as a controlled-controlled NOT, similar to the CNOT gate, but there are two control qubits.
SWAP	Swaps the quantum states of two qubits.
S (Phase Gate)	Adds a phase of $\pi/2$ to the qubit.
T ($\pi/8$ Gate)	Adds a phase of $\pi/4$ to the qubit.
Phase (ϕ Gate)	Applies a phase shift of ϕ to the qubit.
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.

Table 2. Classical gates

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.

3.4. 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. Figure 2 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.

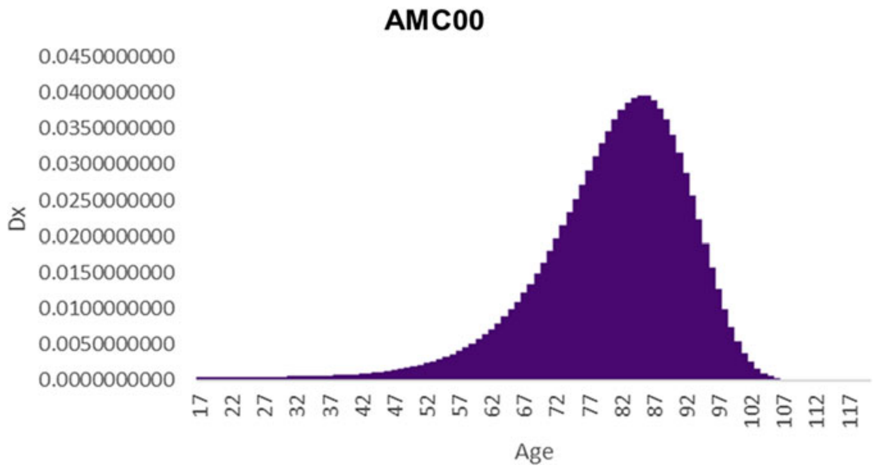


Figure 2. An actuarial ‘life’ table.

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

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

By contrast, the above probability distribution can be captured using just 7 logical qubits (Figure 3 below).

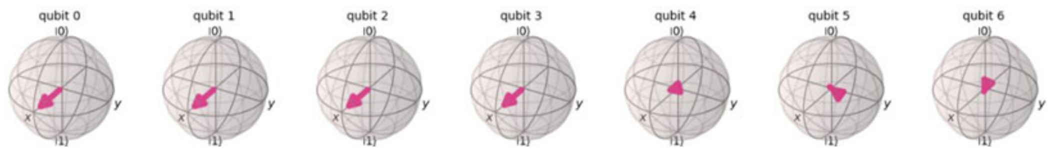


Figure 3. Actuarial ‘life’ table encoded in 7 qubits.

The reason this works is that the 7 qubits span $2^7 = 128$ basis states, with each basis state having a probability attached to it. To encode a life table into 7 qubits, we just need to create a quantum circuit such that the probability of death in any particular age (as observed at birth) corresponds to the probability of finding the 7 qubits in a particular basis state. Note that the last 3 qubits (q4,q5 and q6) are entangled, which means that their state cannot be decomposed in to states of individual qubits. Written in long form the quantum state can be written as:

$$|\Psi> \; = a_{0000000}|0000000> \; + a_{0000001}|0000000> \; + a_{0000010}|0000010> \; + \ldots\ldots\ldots\ldots\ldots$$
$$+ a_{1111110}|1111110> \; + a_{1111111}|1111111>$$

The basis states |0000000>, |0000001> ……… |1111110> and |1111111> are then just binary representations of age (x last birthday) and the coefficients $a_{0000000}$, $a_{0000001}$, ……., $a_{1111110}$, $a_{1111111}$ are just the probability amplitudes representing $\sqrt{Dx}$ for each age. The reader is reminded that in quantum terms the probability of finding the quantum system in a particular state is given by squaring the probability amplitude.

Sampling from the above distribution using a classical computer requires further bits for implementing the Monte-Carlo calcualtion, in the case of quantum computation, we just create the state encoding the above distribution and execute the circuit several times, i.e. we get the randomness for 'free'.

4. Classical versus Quantum Value at Risk

In their seminal work, Egger et al. (2019) highlight the potential of the quantum amplitude estimation (QAE) algorithm to achieve quadratic speedup in risk management applications, such as the calculation of Expected Value, 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 SCR calculations for the Solvency II Internal Model (SII IM) for firms without the use of quantum algorithms (Figure 4).

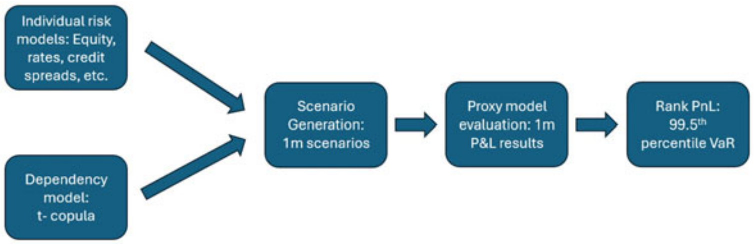

Individual risk models: Equity, rates, credit spreads, etc.
Dependency model: t- copula
Scenario Generation: 1m scenarios
Proxy model evaluation: 1m P&L results
Rank PnL: 99.5th percentile VaR

Figure 4. Schematic of an SCR calculation.

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.

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 P&L 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 P&L.

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 compartmentalisation, 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.

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 delve into the intricacies of each component within the quantum framework, juxtaposing them against their classical analogues, and discussing 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.

5. 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 QAE algorithm. This is a strategic exclusion, as the Internal Model's utility extends beyond merely calculating the 99.5th percentile VaR; it is also utilised for generating the comprehensive distribution of P&L. 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 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 conceptualising quantum computations, stripping away extraneous complexities and presenting the essence of quantum operations. For the development and visualisation of quantum algorithms, we have employed the Qiskit framework, a Python-based library renowned for its utility in quantum programming. The implementations have been rigorously validated on both quantum simulators on a laptop computer, and on those available through the IBM cloud platform (for larger problems).

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

5.1. High Level Quantum IM Calculation

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

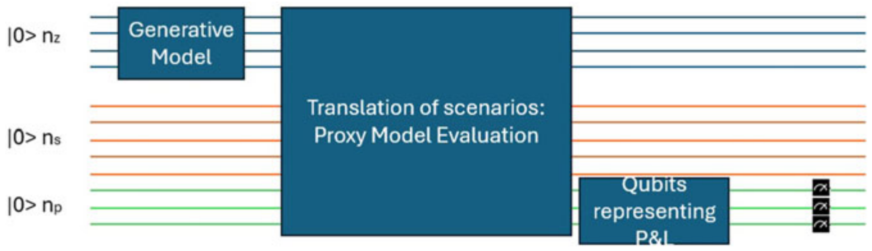


Figure 5. Schematic for a quantum internal model.

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 n_z 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) as set out by Jacquier *et al.* (2022), which is a quantum machine-learning approach that utilises 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 P&L. 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, n_s, depends on the desired granularity of the scenario analysis and the complexity of the proxy model.

Finally, the qubits at the end of the schematic, n_p, are designated for the output – representing the calculated P&L 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 P&L values for each scenario.

The entire process, from the generative model through to the measurement of PL, is conducted within the quantum domain to maintain coherence and leverage the computational advantages of quantum mechanics. The aim is to utilise the inherent parallelism and probabilistic computations of quantum systems to perform risk analysis more efficiently than classical computers. The practical realisation 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 revolutionise the field of financial risk assessment.

5.2. Quantum Generative Model

Figure 6 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.

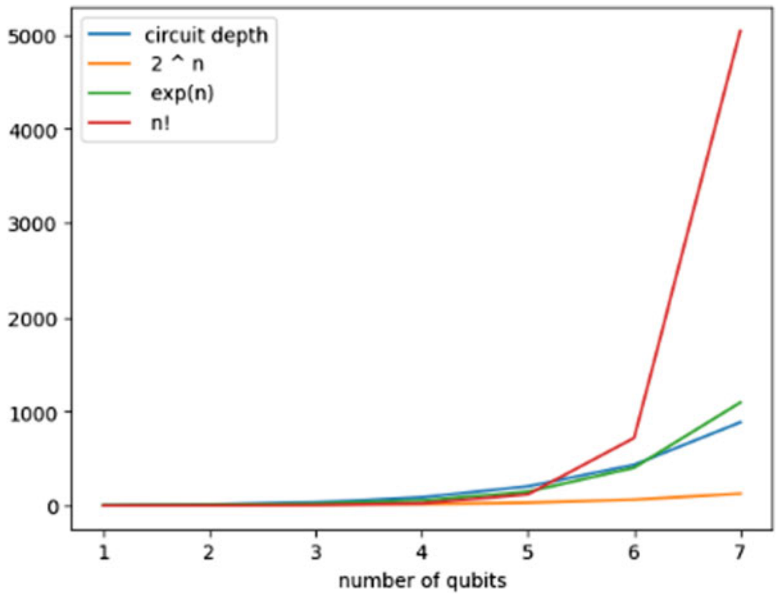


Figure 6. Qubits versus gate counts.

Figure 7 provides an example of an exact implementation for two independent risk factors encoded with 3 (*q0*, *q1* and *q2*) and 2 (*q3*, *q4*) qubits, respectively.

R_y: As illustrated in Figure 1, the state of a qubit can be represented by a position on a Bloch Sphere. All qubits are initialised in state |0>, i.e. the North Pole. This means that upon measurement we will find them in state |0> with 100% probability and state |1> with 0% probability. The R_y gates then represent a rotation around the *Y* axis. The first R_y gate on *q0* represents a rotation of 90 degrees, which changes the state of the qubit from being on the North Pole, to being on the Equator. This changes the probability of distribution of *q0*, as the equator represents an equal super-position of states |0> and |1>, i.e. 50% probability for each.

X: The X gate is the quantum analogue to the classical NOT gate. It flips the state of the qubit. If the qubit is in state |0>, the X gate changes its state to |1>. For qubits in superposition, it 'flips' the probabilities of finding the qubit in the computational basis states (|0> and |1>). If the initial state before applying the X gate, represents a 70% probability of finding q0 in state |0> and 30% probability of |1>, the X gate this to 30% for |0> and 70% for |1>. In bra-ket notation the states before and after the X gate can be written as:

$$|\varphi_0 > \; = \sqrt{0.7}|0> \; + \sqrt{0.3}|1>$$
$$|\varphi_1 > \; = X|\varphi_0 > \; = \sqrt{0.3}|0> \; + \sqrt{0.7}|1>$$

We see that the X gate 'flips' the probability amplitudes associated with |0> and |1>.

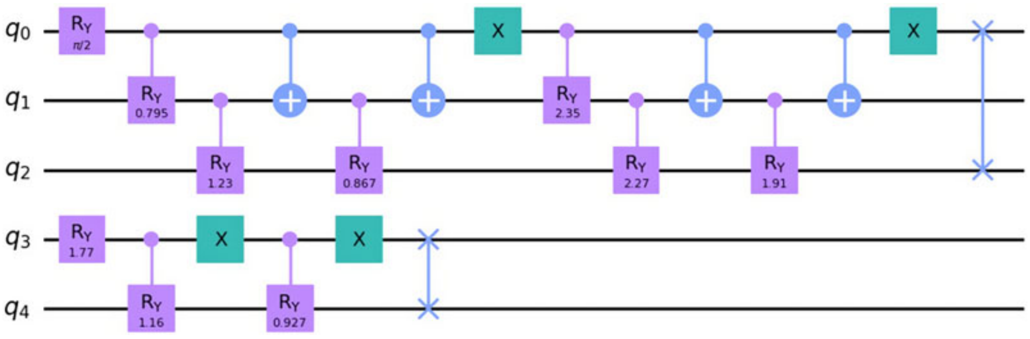


Figure 7. Exact implementation for 2 risks.

CNOT: The light blue circles with a plus sign indicate a controlled-NOT gate (CNOT). This gate flips the target qubit (denoted by the plus sign) if the control qubit (the qubit with the solid circle) is in the state $|1\rangle$. These are used to entangle qubits, creating correlations between them. For example, if *q0* is in an equal super position state, and *q1* is initialised to |0>, applying the CNOT gates conditioned on *q0* will create a combined state such that we will find both *q0* and *q1* in the same state. If we measure *q0* and it is in state |0>, *q1* will be in the same state with 100% probability. Similarly, if *q0* is in state |1>, upon measurement *q1* will also be in state |1> with 100% probability. The 'marginal' probability of measuring *q0* and *q1* in state |0> will be 50%. The reader might recognise that the ability of entanglement to introduce correlations between qubits whilst leaving the marginal probabilities unchanged is similar to copula-based models.

In bra-ket notation the impact of the CNOT can be illustrated as

$$|\varphi_0 > \; = |q0> \otimes |q1> \; = (\sqrt{0.5}|0> \; + \sqrt{0.5}|1>) \otimes |0>$$
$$|\varphi_1 > \; = CNOT[0,1]|\varphi_0 > \; = \sqrt{0.5}|00> \; + \sqrt{0.5}|11>)$$

Note that due to the 'entanglement' the state $|\phi_1>$ cannot be factorised into separate states for *q0* and *q1*.

CRy: The purple boxes connected to solid circles represent the controlled rotations around the y-axis. The action of this operation is more general compared to the CNOT (or *CX*) gate, which rotates the state of the qubit by 180% depending on the state of the control. In the case of *CRy* gates, we can specify how much we want to rotate the target qubit. Whilst the CNOT example above introduced +/- 100% correlations between the two qubits, the *CRy* operation can introduce arbitrary correlations depending on the choice of the rotation angle. A simple worked example is shown below, as before we start with *q0* in an equal super position and *q1* in state |0>.

$$|\varphi_0> \; = |q0> \otimes |q1> \; = (\sqrt{0.5}|0> + \sqrt{0.5}|1>) \otimes |0>$$
$$|\varphi_1> \; = CRy(\pi/2)[0,1]|\varphi_0> \; = \sqrt{0.5}|00> + \sqrt{0.25}|10> + \sqrt{0.25}|11>$$

In simple terms, if *q0* in state |0> leave *q1* unchanged in state |0>. Because *q0* is in equal super position, the probability *q0* is in state |0> is 50%, and this is also the probability that both *q0* and *q1* are in state |0> represented as |00>. In the case q0 is in state |1>, apply a y-rotation on *q1* and put it in an equal super position of |0> and |1>. This means that in the scenarios where *q0* is in state |1>, we will find *q1* in state |0> and |1> with equal probability of 50%. Multiplying these probabilities with 50% of finding *q0* in state |1> we get the overall probabilities:

$$|00> \; = 0.5$$
$$|10> \; = 0.5 * 0.5 = 0.25$$
$$|11> \; = 0.5 * 0.5 = 0.25$$

Returning to the example in Figure 7, the set of quantum gates encodes the probability distributions shown in Figure 8 (below) in an 'exact' way. Note that the exact implementation on 3 qubits (*q0*, *q1* and *q2*) required 13 gates, this increases exponentially with the number of qubits. This is not ideal as 'decoherence' can cause the quantum system to lose information with time, so the longer the circuit the greater the chance that errors will creep in. The QCBM architecture (Figure 9) represents a machine-learning based alternative to the exact implementation, which is much more suited for NISQ devices. This is due to circuit depth that does not increase with the number of qubits.

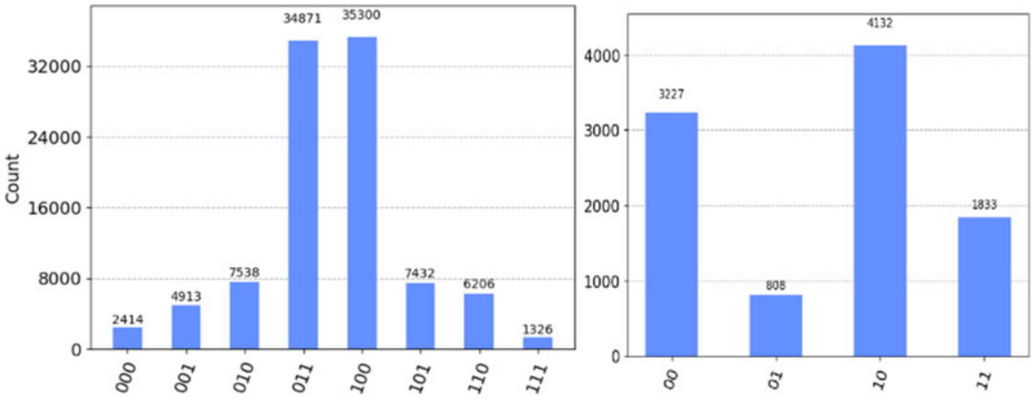


Figure 8. Histograms for the encoded risk factors.

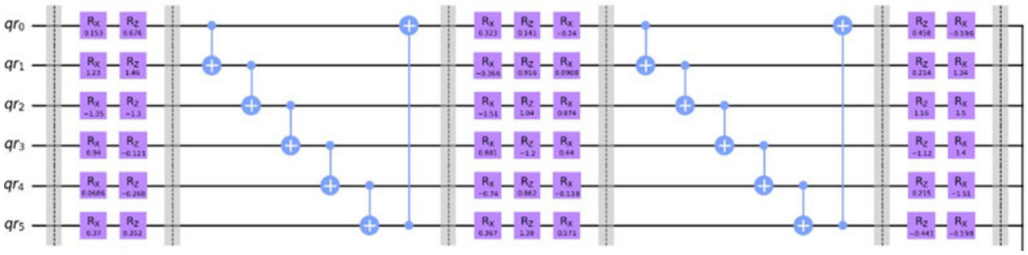


Figure 9. QCBM architecture.

Figure 9 demonstrates the architecture of a QCBM, which is an alternative to 'exact encoding' and virtually has a fixed number of gates per qubit regardless of the number of total qubits. This makes it much more scalable in the context of NISQ devices. This QCBM comprises layers of parameterised rotation gates (*Rx* and *Rz*) and entangling operations (*CX* gates). The fixed architecture suggests a more scalable approach compared to the exact implementation. The rotation angles for each of the gates need to be 'trained' and hence this is usually thought of as a machine-learning based approach 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 per qubit, 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 a computational overhead. 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, like QCBM, Quantum Generative Adversarial Networks (QGANs) can also be 'trained' to create a generative model, with a different architecture, and trade-offs.

5.3. 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.

The proxy model we have chosen for illustration is far from trivial, encompassing a substantial 836 terms to model the NAV directly. Key features are summarised in Table 3:

Table 3. Summary of NAV polynomial complexity

# 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 earlier research. For instance, Egger et al. (2019) detail a two-term polynomial of first order to capture P&L. Their model, denoted by $F(x,y) = 1 - 0.1349x - 0.0186y$, is not only simple but also presents a clear monotonic decrease with respect to the computational basis states.

Building on their previous research, Egger et al. (2020) explore the modelling of a portfolio's loss due to credit defaults, where losses given default (LGD) are exclusively positive integers. 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 we are 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. A more detailed look at the quantum intricacies entailed is as follows:

(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 discretisation. Such granularity 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 the normal distribution onto a quantum scale, ranging from 0 to 1,048,575 in quantum basis states.

The interactions between multiple risk factors require 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.

Each risk factor, encoded with 20 qubits, generates a series of states that must be multiplied with 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 number of 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, 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. We should note that the qubit requirement depends on the exact nature of the proxy model, and that capturing coefficients and the weighted sums requires further qubits, but these additional qubits are marginal compared to the ones required for the affine map.

This quantum proxy model hence extends far beyond the models examined by previous research, which were significantly less demanding in terms of quantum resources (handful of qubits).

5.3.1 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:

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

Here x and y are risk factors represented by 2 qubits each x: [$q0$,$q1$], y:[$q2$,$q3$]

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:

$$x = q0 + 2q1$$
$$y = q2 + 2q3$$

Substituting these relationships into the polynomial F yields:

$$F(q0, q1, q2, q3) = 2 * (q0 + 2 * q1) - (q0 + 2 * q1) * (q2 + 2 * q3)$$
$$+ 3 * (q2 + 2 * q3)^2 = 2 * q0 + 4 * q1 - q0 * q2 - 2 * q0 * q3$$
$$- 2 * q1 * q2 - 4 * q1 * q3 + 3 * (q2^2 + 4 * q2 * q3 + 4 * q3^2)$$

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:

$$F(q0, q1, q2, q3) = 2 * q0 + 4 * q1 - q0 * q2 - 2 * q0 * q3 - 2 * q1 * q2 - 4 * q1 * q3$$
$$+ 3 * q2 + 12 * q2 * q3 + 12 * q3$$
$$= 2 * q0 + 4 * q1 + 3 * q2 + 12 * q3 - q0 * q2 - 2 * q0 * q3 - 2 * q1 * q2$$
$$- 4 * q1 * q3 + 12 * q2 * q3$$

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 benefitted 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:

$$X = -3 + (q0 + 2q1)/(3 - 0) * (3 - (-3))$$

Here, the factor $(3-(-3))/(3-0)$ in the denominator serves to normalise 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 q0 and q1 (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 q0 and q1 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.

5.3.2. 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 Weighted Adder. 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.

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

1. **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.
2. **Decimal encoding:** Alternatively, coefficients could be encoded as decimals within the quantum computing framework.

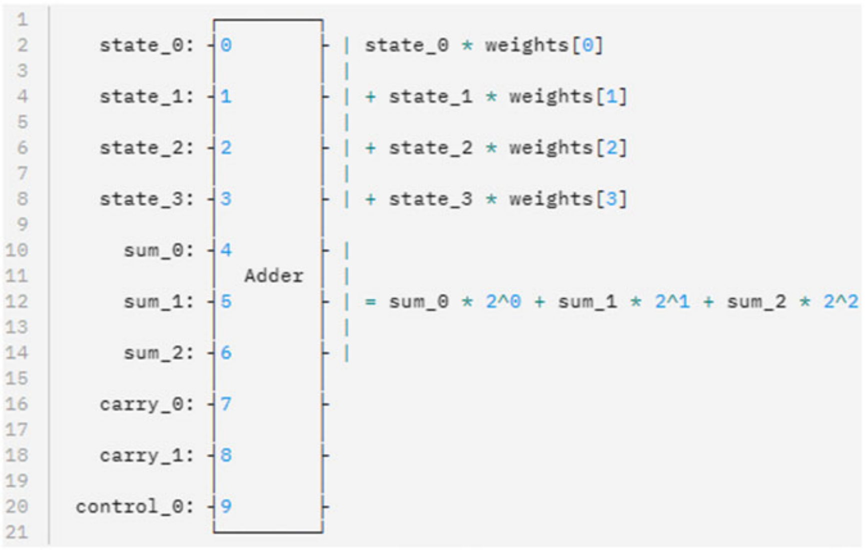
The scaling approach is preferred due to its simplicity and reliability, allowing for high precision while ensuring compatibility with the Weighted Adder's requirements.

Managing positive and negative weights: Since the Weighted Adder 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:

1. **Sum of positive weights:** Aggregate all qubit terms with positive coefficients, applying the Weighted Adder to sum these values.
2. **Sum of negative weights:** Separately, sum all qubit terms associated with negative coefficients. This involves negating the coefficients temporarily to utilise the Weighted Adder.
3. **Combination of sums:** Finally, combine the results from the positive and negative sums. This can be achieved using Two's complement (see Section 5.4.4), which is a mathematical operation on binary numbers that effectively handles subtraction of binary numbers as an addition.

Figure 10 illustrates a high-level schematic of how weighted addition is carried out in the gate based quantum computing framework. Our implementation relies on the IBM Qiskit framework and represents an important part of how polynomials are evaluated on a quantum computer. For example, if a risk factor x is uniformly distribution over the range [0,7], then a simple function $y = 2^*x = 2^*\ (q0+ 2q1 + 4q2)$ can be evaluated using weights (2,4,8) for qubits (q0, q1, q2). Similarly, more complex functions can also be evaluated by choosing appropriate weights. In the Qiskit implementation state_0, state_1 and so on represent the input qubits, whereas the sum is represented by the sum qubits sum_0, sum_1 and so on. While the Weighted Adder 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 is 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.

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 sum qubits. This entails allowing sufficient qubits to hold the outcome of the weighted sum, for example 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:



```
                ┌────────┐
     state_0: ┤0        ├ | state_0 * weights[0]
                │        │ |
     state_1: ┤1        ├ | + state_1 * weights[1]
                │        │ |
     state_2: ┤2        ├ | + state_2 * weights[2]
                │        │ |
     state_3: ┤3        ├ | + state_3 * weights[3]
                │        │
       sum_0: ┤4        ├ |
                │  Adder │ |
       sum_1: ┤5        ├ | = sum_0 * 2^0 + sum_1 * 2^1 + sum_2 * 2^2
                │        │ |
       sum_2: ┤6        ├ |
                │        │
     carry_0: ┤7        ├
                │        │
     carry_1: ┤8        ├
                │        │
   control_0: ┤9        ├
                └────────┘
```

Figure 10. Weighted adder schematic.
Source: Weighted Adder | IBM Quantum Documentation.

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

Here, the floor function represents rounding down to the nearest whole number, and λj is the weight assigned to the j^{th} qubit. It should be noted that the additional qubits required to handle the 'weighted sum' are marginal compared to the c35m qubits for the handling of the multi-qubit interactions in our quantum polynomial construction.

As set out, as the Weighted Adder implementation in Qiskit does not permit negative weights, the positive and negative weights need to be operated on separately and summed together at a later stage. This summation can be carried out by converting both the positive and the negative sums in the respective sum registers to a Two's complement format. The calculation of a Two's complement is explained in Section 5.3.3.1.

5.3.3. Calculating net asset value

The Weighted Adder circuit shown in Figure 10 is utilised to calculate the quantum weighted sum. The sum qubits represent the additional qubits required to hold the output of weighted sum. While we do not know what the weighted sum will be we can calculate the additional qubits required precisely. Following 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.

5.3.3.1. Two's complement. Up to this point, both positive and negative sums are represented by positive values due to Weighted Adder implementation in Qiskit not permitting negative weights, which makes a direct addition 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 Least Significant Bit (LSB) which is the right-most bit 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 Most Significant Bit (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.

5.3.3.2. Ripple carry addition. For the amalgamation of these two registers (post two's complement), the CDKM ripple carry adder can be utilised effectively. This adder requires that both 'sum registers' match in size, necessitating the expansion of the smaller register with additional qubits to ensure compatibility. Other schemes for addition are also available, but the CDKM scheme was chosen because it can carry out the summation with minimum extra qubits.

In Figure 11, the CDKM Ripple Carry Adder's configuration is illustrated, outlining its fundamental structure. Here, cin_0 symbolises 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. This method can be used in the Qiskit framework 'out of box', so a detailed explanation is not provided here. Interested readers are referred to Cuccaro et al. (2004).

The resultant sum through this addition process is effectively captured by four qubits, represented as $|\text{cout}_0, b_2, b_1, b_0>$. The reader is reminded that the registers to be summed need to be of the same size. For example, if the sum over the negative register can be captured by 5 qubits, and the positive weights by 10 qubits (both post two's complement), extra qubits will need to be added to the negative register before carrying out ripple carry addition.

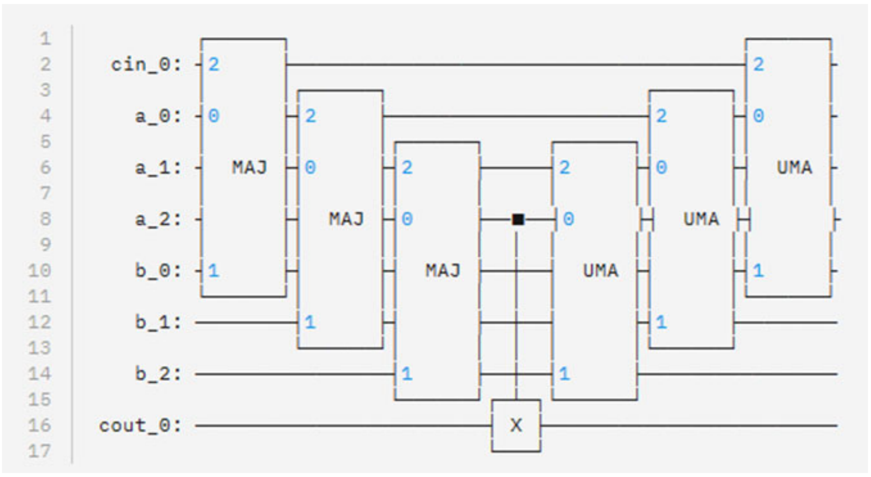


Figure 11. Schematic of CDKM ripple carry adder.
Source: CDKMRippleCarryAdder | IBM Quantum Documentation.

5.3.4. Calculating VaR

Upon completion of the preparatory steps, executing the quantum circuit multiple times serves a purpose similar to that of running simulations in a Monte Carlo method. However, a distinct 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 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 P&L of a company. This distribution is the outcome of the encoded proxy model – a polynomial function mapping the myriad risk variables (such as interest rates, inflation rates, property values, equity prices, and longevity risks) directly to the company's value.

Importantly, the reader is reminded that the output states from the quantum circuit are in a two's complement format. This encoding was chosen to facilitate the handling of both positive and negative values within the binary system. To derive meaningful NAV results from each scenario, these two's complement-encoded outputs must be decoded back into their respective integer values, which can then be converted to a loss amount. This decoding step is crucial for translating the binary data produced by the quantum circuit into meaningful financial outcomes.

5.4. VaR using Quantum Amplitude Estimation

Having explored the direct computation of VaR with quantum computing, we now focus on quantum algorithms that confer a quantum advantage, particularly through QAE.

QAE, as delineated in the foundational work by Egger et al. (2019), 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>. The overall structure of the problem can be written down as follows:

$$A\left|0>_{n+1} = \surd(1-a)\left|\Psi_0>_n\right|0> + \surd(a)\left|\Psi_1>_n\right|1>$$

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

The goal is to estimate the VaR of a portfolio using QAE. 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> 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> with probability a. Otherwise, it will be in the state |0>. The threshold is chosen to match the desired confidence level for the VaR calculation.

For example, for the SCR, we want a threshold corresponding to $a = 0.5\%$, which means that the last qubit will be in the state |1> 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 P&L (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 done such that each of the individual risks is distributed over its computational basis states |0> to $|2^{ni}$-1> where n_i represents the number of qubits encoding risk i.

(2) The proxy model capturing the sensitivity of the Own Funds (or P&L) 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> if the loss exceeds a particular threshold.
(4) Estimate the probability a of the measurement qubit finding itself in state |1>
(5) Carry out a bisection search until a equals the desired confidence level for the VaR calculation.

5.4.1. 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 P&L distribution directly, as set out below:

(1) Measuring just the indicator versus measuring all qubits representing the P&L.
(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 p-value 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 the 1,000,000 typically used, even for $m = 60$.

5.4.2. Building a comparator

The job of the comparator is to flip the state of the measurement qubit, if the P&L exceeds a specified threshold. As the P&L 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. 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 P&L exceeds a certain pre-specified threshold.

This can be achieved by:

(1) Encoding the two's complement of the threshold
(2) Adding this to the PL
(3) If the P&L exceeds the threshold, there will be a carry out on the measurement qubit.

Figure 12 shows the implementation of a comparator for the quantum generative risk models shown in Figure 7. Here the P&L of the system is simply the computational basis state obtained by measuring the qubits $|q_5q_4q_3q_2q_1>$ converted into an integer and can be calculated as q1+2q2+4q3+8q4+16q5 and would span the range [0,31].

The two's complement of the threshold is encoded in qubits $|q_{10}q_9q_8q_7q_6>$, which is 01111(in the example), which means that the threshold is the 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 $|q_5q_4q_3q_2q_1>$ exceeds 17, i.e. when q_5 and at least one other qubit is in state |1>.

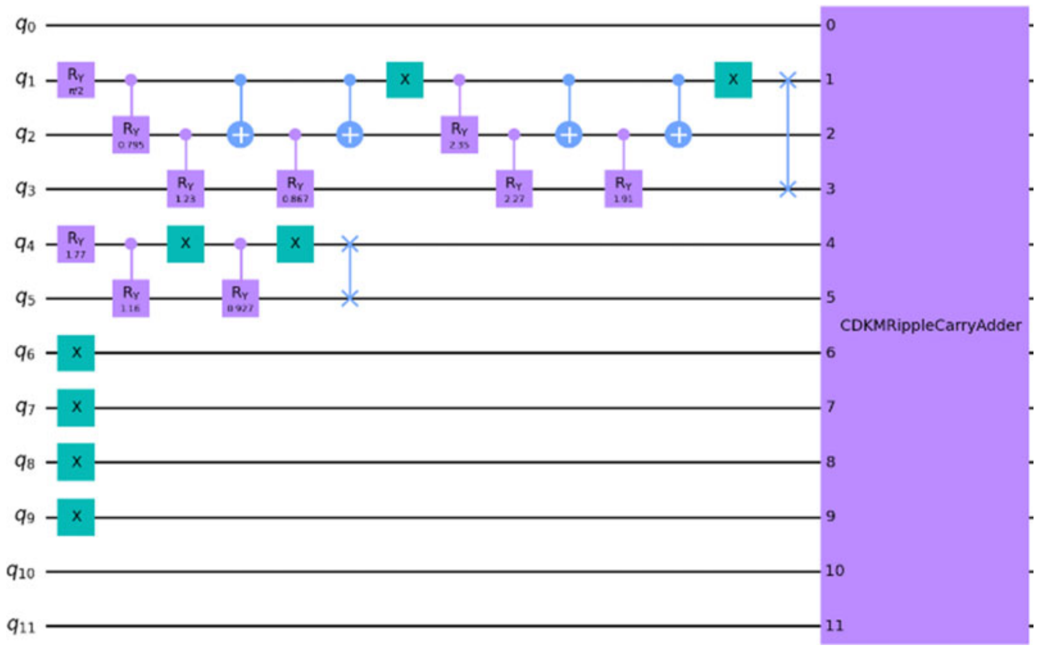


Figure 12. Stylised example of a comparator circuit.

5.4.3. 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>. For VaR calculations, we aim to determine the P&L such that, when it is used as a threshold, the probability of observing the 'measurement qubit' in state |1> 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> 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 P&L 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-monotonic sequence of numbers, because 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 Table 4.

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>. However, due to the non-monotonic ordering of two's complement values, both negative and positive P&L values around this threshold would incorrectly trigger the flip. Specifically, P&L values from -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 P&L exceeds the set threshold (Table 5).

Table 4. Illustration of two's complement

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

Table 5. Modified two's complement

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> only under the appropriate conditions.

Note that the state of the Most Significant Bit (MSB) depends on whether the P&L is positive or negative. This does not require that we explicitly measure the qubits representing the PL, as described in Section 5.4.4. Instead, we can use a controlled-Not gate to flip the state of the extra qubit to |1> if the MSB is |0>, or keep the extra qubit at |0> otherwise.

5.4.4. How to two's complement an 'unobserved' quantity

As mentioned above, the QAE based approach is different to the Monte Carlo based approach where an explicit distribution of P&L is generated. In the QAE case, we ensure that all the risk models, and their relationships to each other and the P&L are encoded in the quantum system, and the quantum advantage arises from estimating the probability of finding a single qubit in state |1> and carrying out a binomial search comparing it to the VaR threshold (e.g. 0.5% for SCR) instead of measuring all the qubits that encode the P&L. Directly measuring the P&L would result in a 'collapse of the quantum state' and the quantum calculation would collapse to its classical counterpart. To preserve the structure of the problem in the quantum state, we circumvent the need to directly measure the P&L for each simulation. We do this by appending a 'comparator circuit' to our implementation which would flip the state of a 'measurement qubit' initialised to |0> if the P&L exceeds a prespecified threshold, as illustrated in Figure 13.

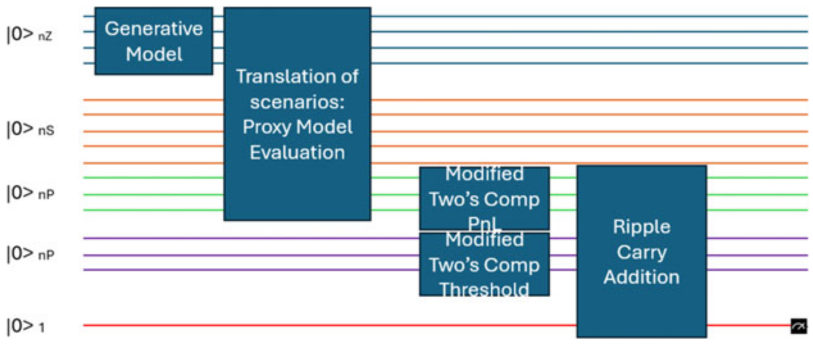


Figure 13. Building the operator A for QAE.

Figure 13 illustrates how the operator *A*, in the context of QAE, is implemented. Notice that, in contrast to Figure 5, there are three further circuits that are implemented: one for calculating the MTC of the PL, one for the MTC of the Threshold and a Ripple Carry Addition Circuit that compares the P&L to the Threshold and flips the state of the final qubit to |1> if the P&L exceeds it.

Whilst we can run this circuit several times to calculate the probability of finding the last qubit in state |1>, in practice the last qubit is not measured. QAE is used to estimate the probability of finding this qubit in state |1>; this crucial step is what gives rise the quadratic speedup as opposed to direct measurement, but it is instructive to show the calculation in this manner.

5.4.5. Quantum amplitude estimation (QAE)

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>. The accuracy of the estimate depends on the number of qubits used for sampling the phase.

Figure 14 shows the schematic for the canonical QAE algorithm (Brassard et al., 2002). At a high level, this algorithm is split into two components, measurement qubits (first register), and the *A* operator (second register). To calculate the SCR, we implement an *A* operator that flips the state of the final qubit to |1> (as per Figure 13), but instead of directly measuring the final qubit and calculating the probability of finding it in state |1> empirically, QAE uses a quantum phenomenon known as phase-kickback to achieve that more efficiently.

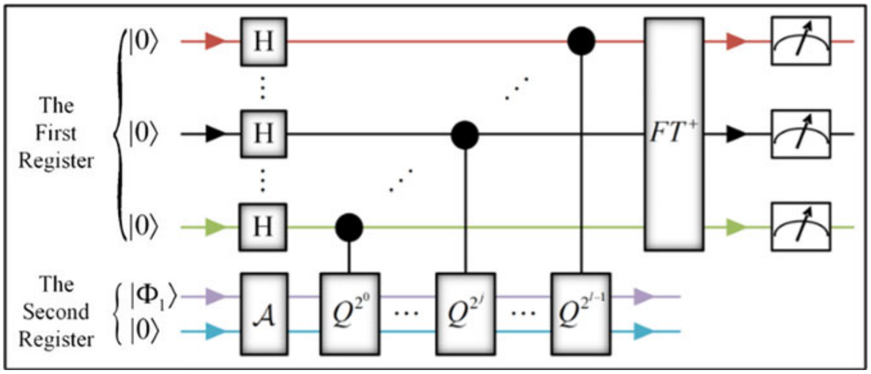


Figure 14. Canonical Quantum Amplitude Estimation. Source: The quantum circuit of the generalised quantum amplitude estimation | Download Scientific Diagram (researchgate.net).

This algorithm has the following distinct components:

The A operator, which we have already covered.

The measurement register (first register in Figure 14), 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, which 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>, 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:

$$P(\varphi, N) = sin(\varphi/(2^N) * \pi/2)^2$$

where Φ is the observation in binary translated into an integer, and 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 (Table 6).

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

The reader might be intrigued as to why we have estimated the probability in such a roundabout way, going through phases and trigonometric functions, rather than finding a way of encoding *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, but 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 (Grinko et al., 2019), Maximum Likelihood Amplitude Estimation (Suzuki et al., 2019), and Faster Amplitude Estimation (Nakaji, 2020). These approaches are available in Qiskit via the AmplitudeEstimator interface.

Table 6. Mapping between measurement outcome and QAE probability

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

5.4.6. Bringing it all together: binary search

Section 5.4.5 describes how the probability amplitude associated with finding the objective qubit in state |1> is encoded as the phase in the First Register. There is 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 the threshold, that results in a probability estimate in line with the VaR confidence interval in at-most N steps.

The quantity we vary in each iteration of the bisection search will be the P&L Threshold. 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 P&L are known a priori, this will not necessarily be the case for the SCR calculations. We sample the P&L directly as in Section 5.3.4 but that will defeat the purpose of using QAE. Even though we do not know the bounds of our P&L function, we do know that the losses will be sorted, thanks to the adjusted two's complement approach in Section 5.4.4.

Also, the SCR calculations from previous cycles provide reasonable guesses for the bounds. We can use the following approach to establish updated bounds before initiating the bisection search.

1. **Start with an initial guess** for the bounds. This can be the maximum and minimum P&L 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 the guessed bounds.
4. **Once suitable bounds are found**, apply the bisection search within these bounds to find the target value.

6. 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 would be able to 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 specialised tasks that are inherently complex for classical computing architectures.

Such tasks might include complex simulations of quantum mechanical systems, optimisation 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 is 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, namely the calculation of SCR for Internal Model firms in Europe/UK. The theoretical foundation was laid out by Egger et al. (2019), 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.

6.1. Why polynomials?

Egger et al. (2019) used polynomials to approximate the P&L of a portfolio for several reasons, which are closely tied to the inherent strengths and operational paradigms of quantum computing, as well as to the specific needs of financial risk analysis:

1. **QAE**: Quantum computing offers unique advantages in certain computational tasks, among which is 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 P&L 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 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 P&L distribution can take various forms depending on the underlying assets and market conditions. By using polynomials, Egger et al. (2019) 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 P&L 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 optimisation 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 optimisation and risk management.

The use of polynomials in both quantum risk analysis and the 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.

6.2. 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.

6.3. 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. (2020) 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. 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 is to consider whether one can speed up the calculation by replacing classical computers with quantum computers for specific parts or the whole calculation.

Figure 15 shows the processes involved in the end-to-end calculation of the SCR for an internal model firm. The dotted box shows the components of the calculation that we have identified for quantum acceleration.

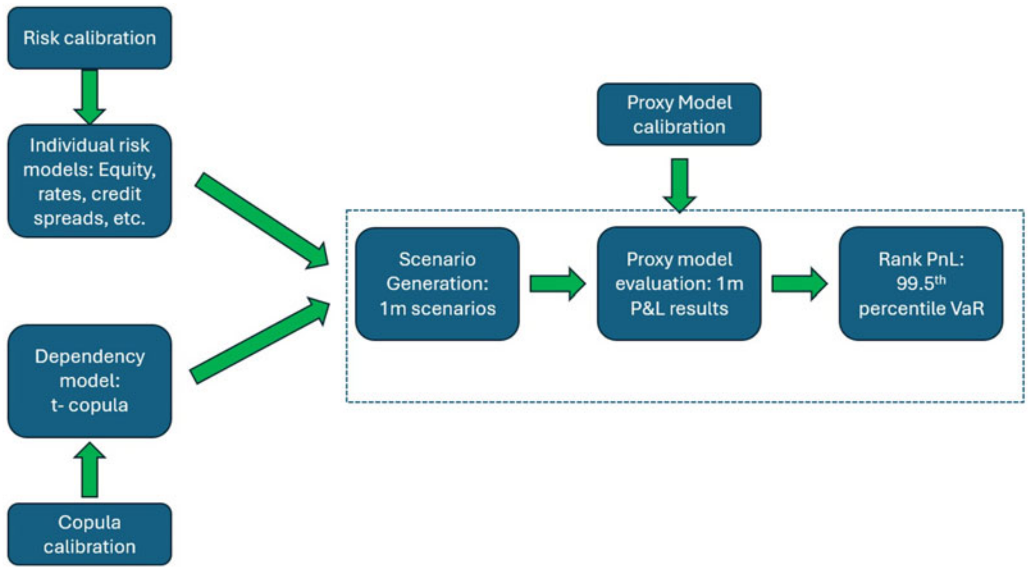

Risk calibration
Individual risk models: Equity, rates, credit spreads, etc.
Dependency model: t- copula
Copula calibration
Proxy Model calibration
Scenario Generation: 1m scenarios
Proxy model evaluation: 1m P&L results
Rank PnL: 99.5th percentile VaR

Figure 15. End to end SCR calculation.

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'. Figure 16 (below) illustrates the process 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.

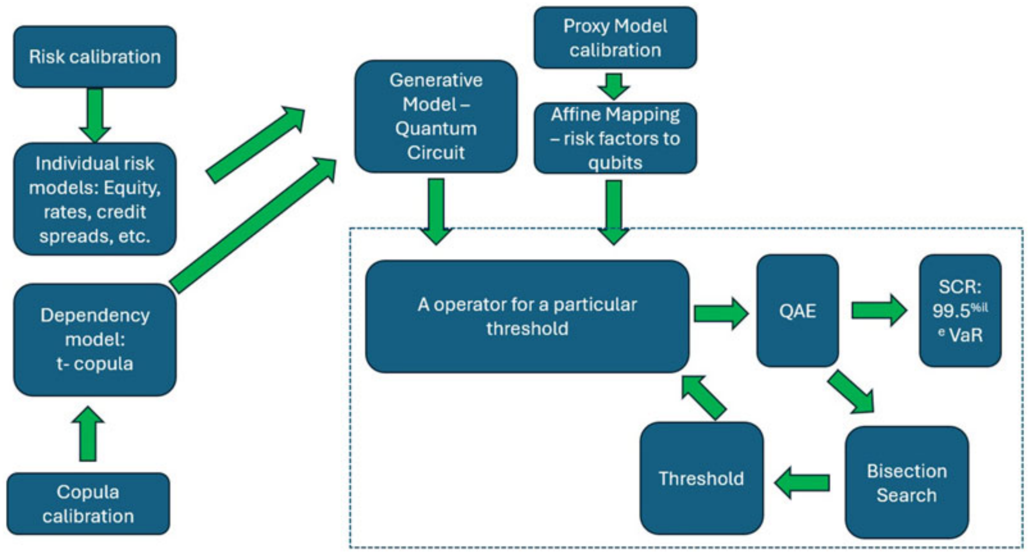

Risk calibration
Individual risk models: Equity, rates, credit spreads, etc.
Dependency model: t- copula
Copula calibration
Generative Model – Quantum Circuit
Proxy Model calibration
Affine Mapping – risk factors to qubits
A operator for a particular threshold
QAE
SCR: 99.5%ile VaR
Threshold
Bisection Search

Figure 16. SCR calculation with quantum acceleration.

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 initialisation.
   (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 5.3.1, it is outlined that proxy models typically function by mapping the values of risk variables directly to P&L. However, a direct application of these models is not feasible due to the necessity of translating these mappings into a form compatible with 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, $|q2q1q0\rangle$, in a range [0,7], would be transformed into $x = q0+2q1+4q2$ via affine mapping. This transformation process, however, is notably complex, with different strategies presenting their own advantages and challenges:
      - **Computational graph construction or symbolic computing**: 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 5.3.1 illustrates the mapping from risk factor polynomial to a qubit polynomial, reproduced in Tables 7 and 8.

Table 7. Illustration of NAV as a function of risk factors

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

Table 8. NAV as a function of qubits (where qubits encode underlying risk distributions)

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 in Table 8 are straightforward, terms 5-9 can be constructed via multi-controlled x gates targeted at qubits initialised to $|0>$ - 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 5.3.2.

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 initialisation 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.

6.4. 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 an 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 several SCR calculations could potentially be executed on a classical computer. This overhead is due to the translation of the Monte-Carlo problem in quantum terms and the construction of the quantum circuit, both of which are carried out on classical computers. In simpler terms, the performance benefit of running the SCR calculation on a quantum computer is far outweighed by the overhead necessary to translate the problem from classical to quantum terms.

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.

6.5. 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 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 initialisation 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. QEC schemes address the challenge of limited gate fidelity in the following way:

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 versus logical qubits**: It is 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 feasible, and protocols like the surface code are designed to maximise 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 approximately 36 million logical qubits, which means that roughly 36 billion physical qubits might 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 minutes/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.

6.6. The Quantum Conundrum: Is It Worth It?

Sections 6.1–6.5 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 and part two is more general.

Quantum risk appetite

While it is true that 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, demonstrating they meet the 'use-test' criteria.

Companies often define their risk tolerances and constraints according to the calibrations of their internal models. 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 utilised 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 Figure 17.

The chart displays the Solvency Ratio (SR) distribution for a hypothetical company, with the solvency ratio calculated as:

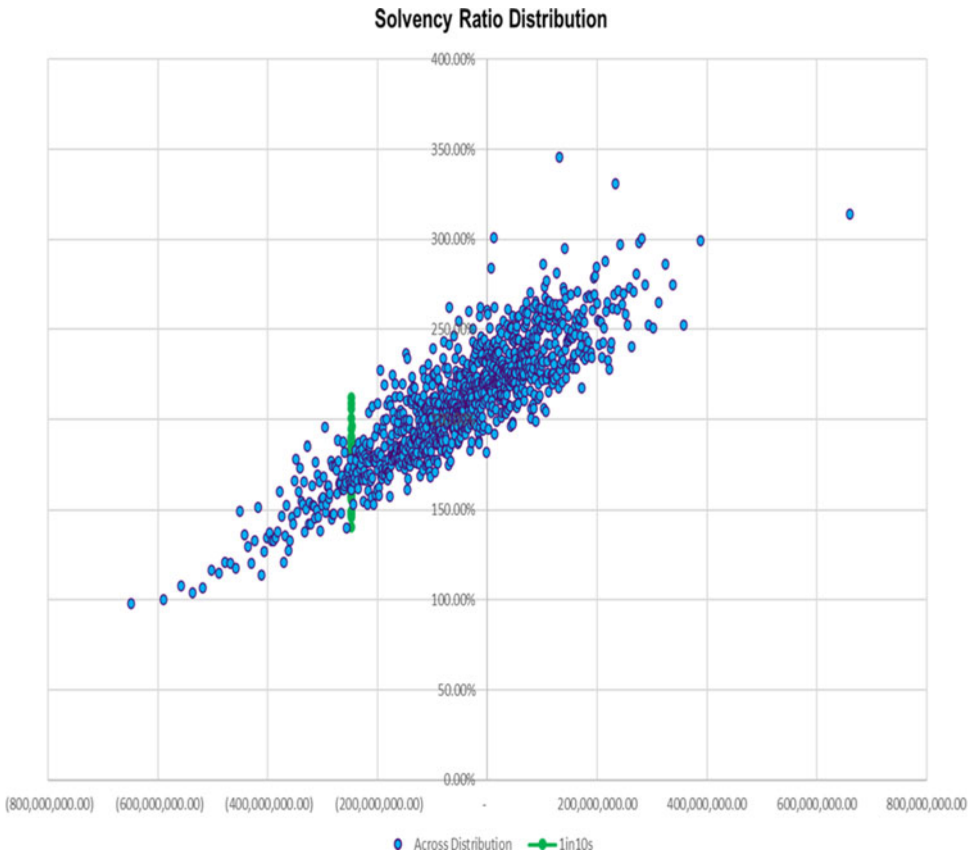


Figure 17. Solvency ratio distribution.

$$SR = NAV/SCR$$

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 NAV 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 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 is 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 realise 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 that are 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, optimising large-scale logistics and supply chains, and providing new insights into natural phenomena through simulation.
3. **Advancing cryptography**: Quantum computers could revolutionise 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 organisations 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 optimise energy consumption, waste reduction, and resource allocation, contributing to more sustainable solutions and helping tackle climate change.

6.7. Avenues for Future Research on this Topic

The preceding sections have critically examined the formidable challenges associated with harnessing quantum advantage for 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 prioritise 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 NAV or P&L – 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 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.

7. Conclusions

This paper critically assesses the potential for 'quantum advantage' in risk management applications, such as the calculation of 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.

7.1. 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.

7.2. 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 (potentially 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 imposes 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 realising 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

Berry, T. & Sharpe, J. (2021). Asset–liability modelling in the quantum era. *British Actuarial Journal*, **26**, e7. https://doi.org/10.1017/S1357321721000076

Brassard, G., Høyer, P., Mosca, M. & Tapp, A. (2002). Quantum amplitude amplification and estimation. *Contemporary Mathematics*, **305**, 53–74. https://doi.org/10.1090/conm/305

Cuccaro, S.A., Draper, T.G., Kutin, S.A. & Moulton, D.P. (2004). A new quantum ripple-carry addition circuit. *arXiv preprint* arXiv/0410184v1.
Egger, D.J., Gutiérrez, R.M., Mestre, J.C. and Woerner, S. (2019). Quantum risk management. *Quantum Finance*, **1**, 105–112.
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. https://doi.org/10.1109/TQE.2020.3030314
Grinko, D., Gacon, J., Zoufal, C. & Woerner, S. (2019). Iterative quantum amplitude estimation. *arXiv preprint* arXiv:1912.05559.
Jacquier, A., Kondratyev, O., Lipton, A. & López de Prado, M. (2022). *Quantum Machine-learning and Optimisation in Finance*. Packt Publishing.
McNeil, A.J., Frey, R., and Embrechts, P. (2015). *Quantitative Risk Management: Concepts, Techniques and Tools*. Revised Edition. Princeton University Press.
Nakaji, K. (2020). 'Faster Amplitude Estimation', *arXiv preprint* arXiv:2003.02417.
Nielsen, M.A. and Chuang, I.L. (2010). *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press.
Preskill, J. (2018). Quantum computing in the NISQ era and beyond. *Quantum*, **2**, 79. https://doi.org/10.22331/q-2018-08-06-79
Rudin, W. (1976). *Principles of Mathematical Analysis* (3rd ed.). McGraw-Hill.
Suzuki, Y., Uno, S., Raymond, R., Tanaka, T., Onodera, T. & Yamamoto, N. (2019). Amplitude Estimation without Phase Estimation. *arXiv preprint* arXiv:1904.10246.

Cite this article: Amjad MA. (2025). Quantum internal models for Solvency II and quantitative risk management. *British Actuarial Journal*. https://doi.org/10.1017/S135732172400031X