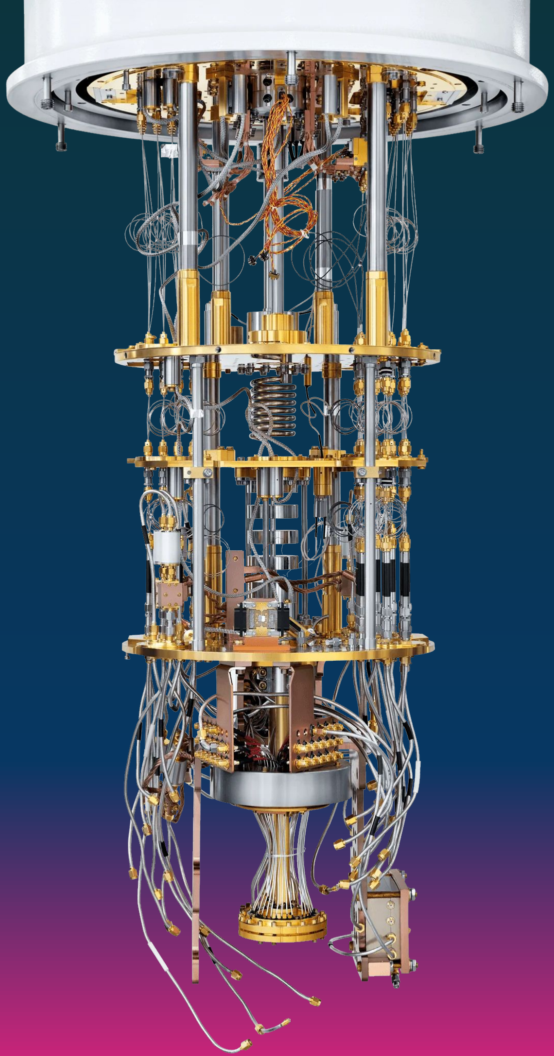
Quantum Computing for Financial Risk Analysis:

Pricing Fixed-Income Assets with *QuMonte*



Sophia Rebello
Rice University

March 4, 2025

RICE

About Me

- Senior at Rice University; Computer Science & Finance major
- Originally from Houston, TX
- **Fun Fact**: Competed on Food Network's *Chopped Junior* in 7th grade (Season 4 Episode 1 Cup of Glee) & cooked for celebrity judge Meghan Markle

Professional Experience

- Citi – Quantitative Analysis Intern (Central Risk Team)
- Mako Trading – Quantitative Research Intern
- Chevron – Software Engineer Intern/Finance Intern

Why Financial Risk Analysis?

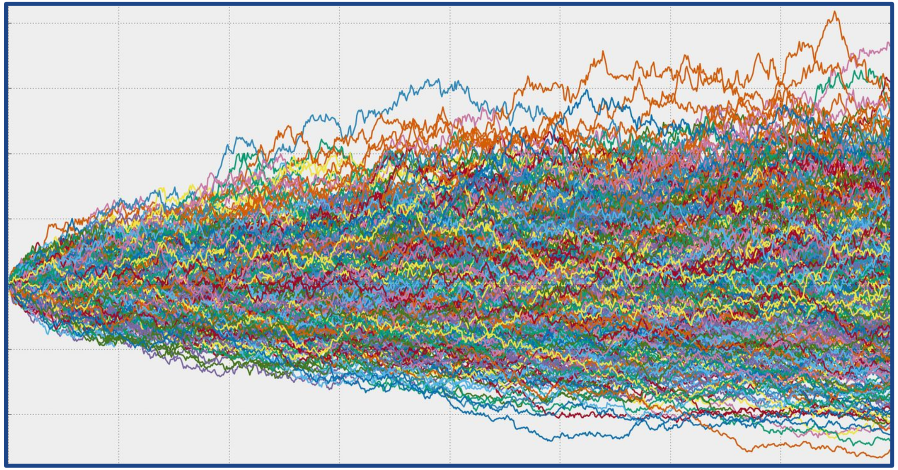
- Passion for quantitative finance sparked by internships in trading, risk modeling, & algorithmic analysis
- Experience in Value-at-Risk modeling, Monte Carlo simulations, & factor risk led me to explore quantum computing's potential in finance
- Quantum finance use cases → Risk assessment, fraud detection, & portfolio optimization

What is Quantum Computing?

- Processes information up to *exponentially* faster than classical computers
- **Qubits:** Bits of information that can exist in multiple states at once
- **Quantum gates:** Manipulate qubits, changing their state probabilities
  - Pauli (X, Y, Z): Flip/rotate qubits
  - Hadamard (H): Creates superposition
  - CNOT: Entangles qubits
  - Rotation (Rx, Rz): Fine-tune states
- **Quantum circuits/code:** Sequences of gates leveraging parallelism and amplitude amplification

What are Monte Carlo Simulations?

- Model random variables to estimate probabilistic outcomes
- Used in finance to price bonds & mutual funds by simulating future cash flows
- Essential for fixed-income pricing, factoring in interest rates, credit risk, & reinvestment rates

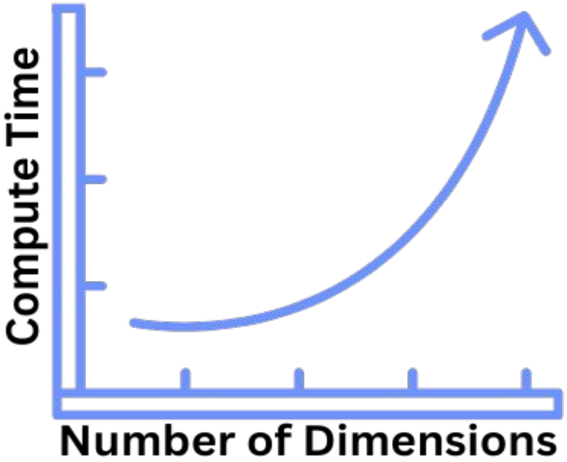


Motivation

The Problem

Finance industry needs fast, scalable, and precise risk assessment, but existing Monte Carlo methods:

- Are costly as simulations struggle to scale
- Face bottlenecks even with GPUs
- Sacrifice accuracy due to computational limits of classical methods



Bridging the Gap

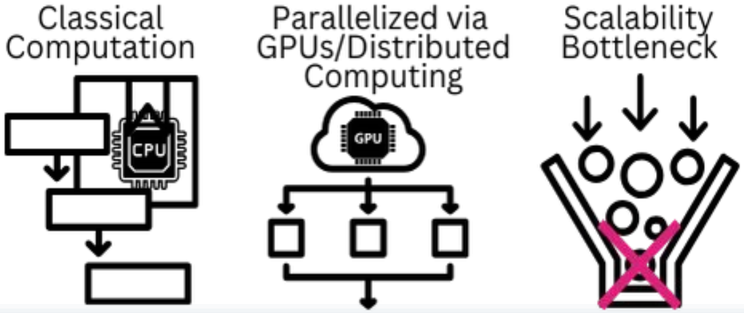
Our Solution

Quantum computing offers a breakthrough by processing multiple scenarios simultaneously, leading to:

- Higher accuracy with optimized probability estimation algorithms
- Reduced computational overhead through optimized quantum code
- Large speedup in Monte Carlo simulations

Challenge

Require efficient and optimized quantum code to achieve this on current noisy and small-scale quantum computers



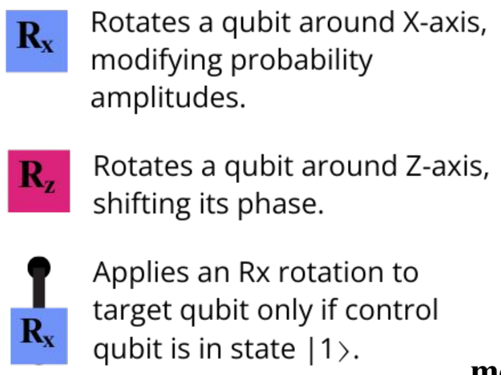
Our Approach

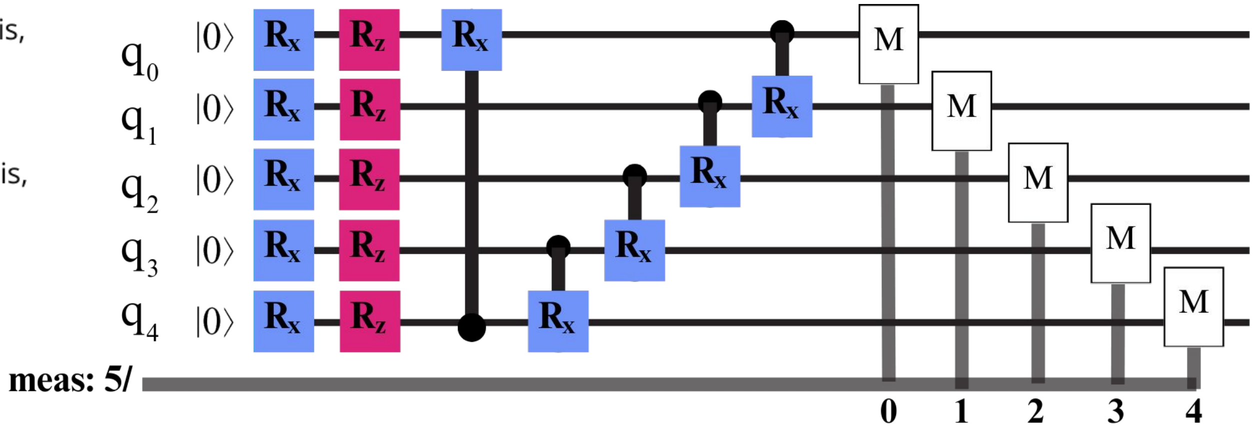
- **Optimized Quantum Code**: Utilize Qiskit language to refine quantum code for IBM's architecture, reducing noise, and code size
- **Variational Quantum Algorithms (VQAs)**: Minimize the L_2 norm, (Euclidean distance) between circuit output and a target financial probability distribution
- **Efficient Optimization**: Employed BOBYQA and other optimizers to achieve high accuracy with fewer computational steps

```
q = QuantumRegister(2,'q')
c = ClassicalRegister(2,'c')

def firstBellState():
    circuit = QuantumCircuit(q,c)

    circuit.h(q[0]) # Hadamard gate
    circuit.cx(q[0],q[1]) # CNOT gate
    circuit.measure(q,c) # Qubit Measurment

    print(circuit)

    job = execute(circuit, backend, shots=8192)

    job_monitor(job)
    counts = job.result().get_counts()

    print(counts)
```

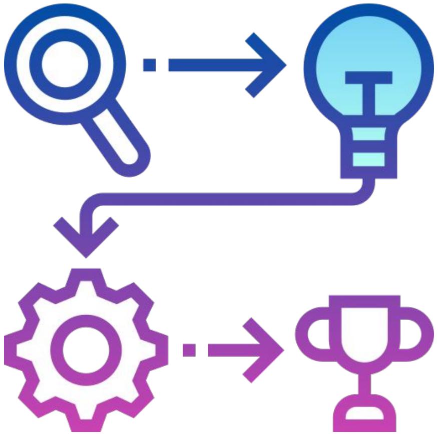
Design & Implementation

- **Quantum Circuit Design:** 5-qubit circuit with Rx & Rz gates is transpiled to native gate set to reduce complexity
- **Optimization:** VQAs iteratively tune θ_1 to θ_{10} using the BOBYQA optimizer to minimize the L_2 norm





Methodology



Simulation & Testing

Evaluate performance through Aer simulations & IBM hardware tests

Metrics

Accuracy assessed via L_2 norm, with Qiskit generating outcome histograms for conditional probabilities across all 32 states

Fixed Income Simulation Model

$$V = \sum_{t=1}^{T} \frac{c_t}{(1+r_t)^t}$$

$$\{0, \ldots, 2^{n_j}-1\} \ni i_j \mapsto \frac{high_j - low_j}{2^{n_j}-1} * i_j + low_j \in [low_j, high_j]$$

Results and Analysis

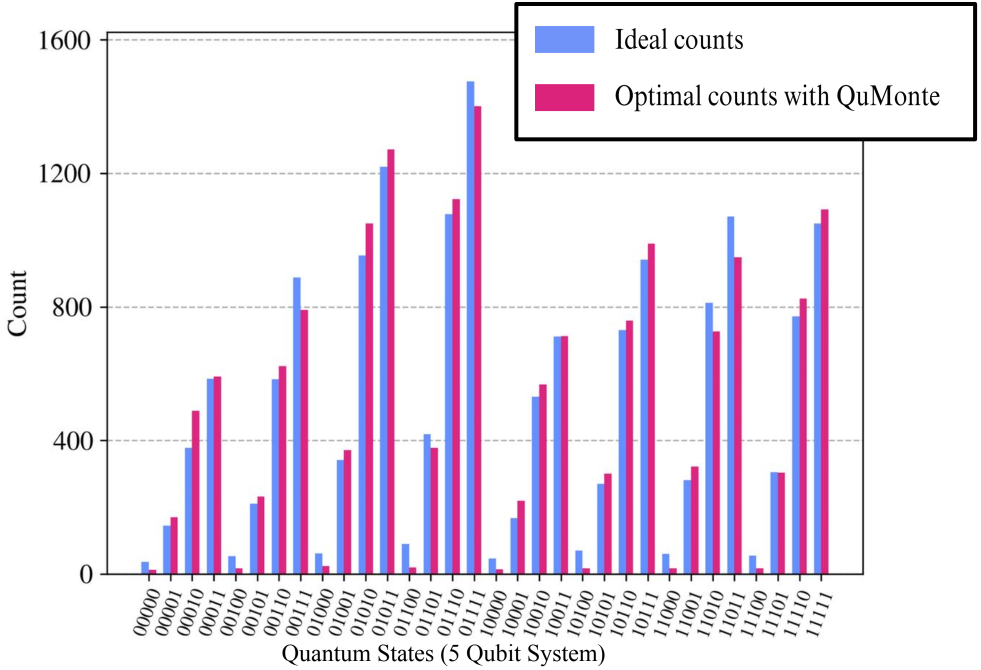
- BOBYQA achieves the lowest L_2 norm (0.0071 @ 4000 iterations), outperforming other optimizers in accuracy and efficiency
- DIRECT-L-RAND was the next best optimizer

Maxiter	DIRECT_L_RAND	COBYLA	NELDER_MEAD	Gradient Descent	BOBYQA
2000	0.0635	0.0176	0.1429	0.0345	0.0113
3000	0.0888	0.0795	0.0537	0.0587	0.011
4000	0.0078	0.0269	0.1952	0.3034	**0.0071**
5000	0.0192	0.3759	0.2027	0.4015	0.0133

Results and Analysis

- *QuMonte*'s optimized circuit closely matches ideal distributions, demonstrating effectiveness in fine-tuning circuit parameters

- Quantum amplitude estimation significantly reduces computational errors in Monte Carlo simulations



Results and Analysis

Why is *QuMonte* superior to existing Monte Carlo methods?

Metric	QuMonte (BOBYQA)	Other Methods
Convergence Speed	Fastest (L_2 = 0.0071 @ 4000 iterations)	Slower (higher L_2 even after 4000+ iterations)
Circuit Depth	Low (5 qubits, single-qubit Rx/Rz gates)	Deeper circuits with multi-qubit CNOT gates, leading to higher noise, increased error rates, & longer execution
Computational Cost	Lower (fewer iterations to converge, fewer qubits, & lower gate complexity)	Higher (more qubits, iterations, & complex gate operations)

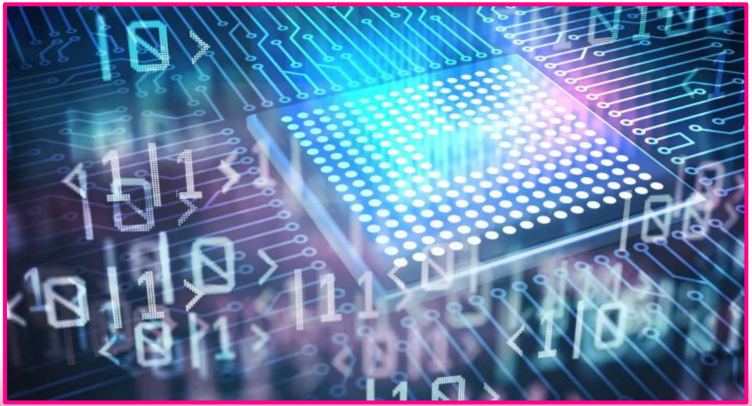
With an optimization score near 0.01(1% error), *QuMonte* enhances financial model precision & efficiency while reducing computational cost.

Our Contribution

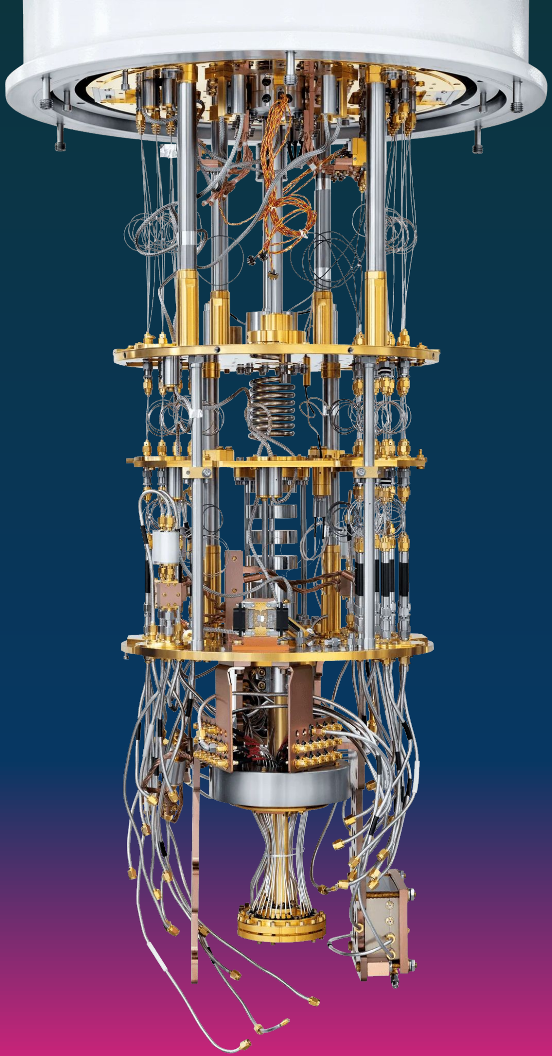
- Circuits optimized with *QuMonte*'s BOBYQA engine reduce quantum noise, depth, & computational complexity for higher accuracy

- *QuMonte* provides a blueprint for applying architectural principles to develop scalable quantum software in financial risk assessment

- Future work will refine the quantum ansatz, optimize hyperparameters and circuit measurements to enhance computation speed, and extend applications to basket and European Call/Put options



QuMonte is Open-Sourced!



Contact info:
sir5@rice.edu

RICE