

Hamiltonian Simulations of Spin Systems by Ising Model

By

Muhammad Mudassar

1457-BS-PHY-21

SESSION 2021-25



DEPARTMENT OF PHYSICS

GOVERNMENT COLLEGE UNIVERSITY, LAHORE, PAKISTAN

DECLARATION

I, Muhammad Mudassar, hereby declare that this thesis titled “Hamiltonian Simulations of Spin Systems by Ising Model” is my work conducted in pursuit of a BS Honors degree at Government College University, Lahore. I carried out this work solely for the BS Honors program. Any assistance or published work of others that I have utilized is clearly acknowledged and cited. All quotations from the work of others are properly referenced, and except for such quotations, this thesis represents entirely my own research. Collaborative work with my supervisor has been clearly indicated, specifying the contributions made by others and those made by myself.

Dated

Signature

CERTIFICATE

The thesis entitled '**Hamiltonian Simulations of Spin Systems by Ising Model**' has been carried out by **Muhammad Mudassar, Roll No. (1457-BS-PHY-21), Session 2021-2025** at the Department of Physics, Government College University, Lahore, Pakistan, under my supervision in partial fulfillment of the requirement for the award of the Bachelor of Science (BS) Honors in Physics. The matter embodied in this thesis is original and has not been submitted for the award of any other degree.

Date:

Supervisor:

Dr. Atif Shahbaz

Assistant Professor

Department of Physics

Govt. College University,
Lahore

Submitted Through:

Prof. Dr. M.N.S. Qureshi

Chairperson, Department of Physics

Govt. College University, Lahore

Controller of Examinations

Govt. College University,
Lahore

DEDICATION

This thesis is dedicated to my beloved parents, whose unwavering love, support, and encouragement have been my guiding light. To my family, for their endless sacrifices and belief in my dreams, and to my sister and brother, for always being my source of strength and inspiration. To my Physics teacher, Dr. Atif Shahbaz, Dr. Fatima Binta Munir and Dr. Sajid Ali for his unconditional support and guidance.

ACKNOWLEDGMENT

In the Name of Allah, the Beneficent, the Merciful

All praise and thanks be to Almighty Allah, whose countless blessings, guidance, and mercy enabled me to complete this thesis successfully. I am deeply grateful for the strength and patience He granted me throughout this academic journey.

I would like to express my sincere appreciation to my supervisor, **Dr. Atif Shahbaz**, for their valuable guidance, continuous support, and encouragement throughout this work.

I am also deeply grateful to **Dr. Fatima Binta Munir** for her expert mentorship, valuable suggestions, and constant encouragement. Her guidance not only helped me improve the technical quality of this research but also inspired me to explore deeper into the field of quantum computing.

I am also extremely thankful to the Head of the Department, **Prof. Dr. M. N. S. Qureshi**, for providing a supportive academic environment and for his invaluable leadership at the Department of Physics, Government College University, Lahore.

My heartfelt gratitude also goes to my parents, my elder brothers **Muhammad Rizwan** and **Muhammad Abdullah**, my sisters and friends for their unwavering love, motivation, and moral support during the most challenging phases of this thesis.

May Allah reward everyone who contributed to my journey in any way. May He guide us all on the path of knowledge and success.

[Muhammad Mudassar]

Abstract

This thesis investigates the quantum simulation of spin systems by focusing on the quasi-one-dimensional magnetic material CoNb₂O₆, modeled through the transverse field Ising model (TFIM). Spin chains provide a valuable framework for understanding critical behavior and correlated quantum phases in condensed matter systems. The TFIM, in particular, offers an analytically and numerically tractable model to study phenomena such as quantum phase transitions, spin excitations, and entanglement scaling. By aligning CoNb₂O₆ with the 1D Ising model under a transverse magnetic field, this study explores the system's low-energy excitations and dynamic behavior using quantum simulation techniques.

The Hamiltonian describing CoNb₂O₆ is formulated using parameters drawn from experimental studies, incorporating both spin-spin coupling and transverse field contributions. Using the Suzuki–Trotter decomposition, the model is encoded into quantum circuits and simulated via Qiskit 2.0, utilizing both noiseless backends and real IBM Quantum hardware (e.g., *ibm_brisbane*). Simulations target the evolution of spin states and the emergence of bound excitations, revealing how quantum fluctuations govern the system near its critical point. Classical results are compared with quantum simulations to evaluate the accuracy, circuit depth, and limitations of current quantum processors.

The findings support the viability of using present-day quantum devices to replicate core features of realistic spin models. This work demonstrates how quantum computation can be applied to simulate nontrivial material systems even before reaching the era of fault-tolerant quantum computers, thereby contributing to the broader effort of bridging condensed matter theory with quantum algorithm development.

Contents

List of Figures	xii
List of Tables	xiii
1 Introduction to Quantum Computing	1
1.1 Quantum Computing Principles	1
1.2 Bit Vs Qubit	2
1.3 Quantum Hardware vs Software	3
1.4 Fundamental Concepts	5
1.5 Bloch Sphere Representation	5
1.6 Quantum Computing Architecture	8
1.7 Quantum Algorithms & Use-Cases	9
1.8 Quantum Computing Roadmaps	13
1.9 Error and Decoherence in Quantum systems	15
2 Quantum Computing Literature	16
2.1 Quantum Gates Representation	16

2.1.1	Types of Quantum Gates	18
2.1.2	Physical Relevance:	19
2.1.3	Unitary Transformations	19
2.2	Fields in Quantum Computing	21
2.3	Analytical vs Numerical (Classical) Simulations	22
2.4	Motivation for Quantum Hamiltonian Simulations	24
2.5	Shifting to Pauli Matrices	25
2.6	Building Quantum Circuits	26
2.7	Methods and Technologies for Quantum Information Processing	26
2.8	Simulation Workflow	27
3	Programming with Qiskit and Jupyter Notebook	29
3.1	Overview	29
3.2	Development Environment Setup	29
3.2.1	Conda Environment Using Miniconda	29
3.2.2	Packages Installed in System Command Prompt	30
3.3	Qiskit Versioning and Practical Considerations	31
3.4	Problems Encountered and Mitigation Strategy	31
3.5	Qiskit Primitives Used	32
3.5.1	Estimator Primitive	32
3.5.2	Sampler Primitive	32

3.5.3	Related Imports	32
3.6	Unitary Gate Utility	32
3.7	IBM Cloud Access History	33
3.8	Qiskit Circuit Transpilation and Measurement Bits	33
3.8.1	Declaring Classical Bits in Quantum Circuits	33
3.8.2	Transpilation and Logical to Physical Qubit Mapping	33
3.8.3	Comparison of SamplerV2 and EstimatorV2 Primitives	34
4	Spin Systems	35
4.1	Spin Systems in Quantum Mechanics and Quantum Computing	35
4.1.1	Quantum Spin: Fundamental Concepts	35
4.2	Spin Lattice Models	36
4.2.1	The Ising Model	36
4.2.2	The Heisenberg Model	37
4.3	Classical vs Quantum Ising Model	38
4.3.1	Classical Ising Model	38
4.3.2	Quantum Ising Model	39
4.3.3	Key Differences	39
4.3.4	Implications for Quantum Computing	40
4.4	Quantum Simulation Techniques	40
4.5	Machine Learning for Spin Models	41

4.6	Experimental Realizations	41
4.7	Transverse Field Ising Model and Custom Unitary Gates	42
4.7.1	1D Transverse Field Ising Model (TFIM)	42
4.7.2	Custom Unitary Gates in Qiskit	42
4.7.3	The Pauli-Y Gate	43
4.7.4	The ZZ Interaction Gate and Basis Rotation	43
5	Hamiltonian Simulation of CoNb_2O_6	45
5.1	Define the Physical System (1D TFIM)	45
5.2	Wave Function Representation	46
5.3	Quantum Ising Hamiltonian for CoNb_2O_6	46
5.3.1	Limiting Cases of the Transverse Field Ising Model	47
5.3.2	Fundamental Criteria for realizing the Ising Model	48
5.4	Hamiltonian Decomposition	49
5.5	Mapping to Pauli Operators	50
5.6	Applying Suzuki-Trotter Decomposition	50
5.7	Mapping to Basic Quantum Gates	51
5.8	Quantum Circuit Construction	51
5.9	Simulation on Real IBM Quantum Computers Using Qiskit	52
5.10	Comparison of Classical and Quantum Simulation Results	52
A	Appendix: Qiskit Implementation and Python Code	57

A.1 Bell State	57
--------------------------	----

List of Figures

1.1	Comparing classical and quantum computing	4
1.2	Bloch sphere, a geometrical representation of a two-level quantum system.	6
2.1	Bell State	17

List of Tables

1.1	Comparison between Classical Bits and Quantum Qubits	3
1.2	Important Quantum Algorithms Overview	12
1.3	Comparative Overview of Quantum Computing Roadmaps and Milestones	14
2.1	"Types of Simulation Methods" A diverse approach used to model, analyze and predict the behaviour of complex systems	24
3.1	Comparison between SamplerV2 and EstimatorV2 Primitives	34
4.1	Comparison of classical and quantum Ising models	39
5.1	Essential Physical Properties for Ising Model Systems	49

Chapter 1

Introduction to Quantum Computing

1.1 Quantum Computing Principles

Quantum computing harnesses the principles of quantum mechanics to process information in fundamentally new ways. Unlike classical systems that operate on bits, which can exist in one of two states (0 or 1), quantum systems utilize qubits, which use the phenomena of superposition and entanglement to carry out calculations.

The superposition allows a qubit to exist in a combination of multiple states simultaneously, rather than being confined to a single binary value. Entanglement, on the other hand, is a quantum property that links qubits such that the state of one qubit instantly affects the state of other, irrespective of the gap setting apart them. Quantum computing is not merely a faster alternative to classical computing; it represents an entirely different computational paradigm. Quantum algorithms, such as Shor's algorithm for integer factorization and Grover's algorithm for unstructured search which are discussed in section 1.4; demonstrates exponential and quadratic speed-ups over their best-known classical counterparts.

These computational advantages suggest that quantum computers have the po-

tential to efficiently solve specific classes of problems that are considered intractable for classical machines.[17]

1.2 Bit Vs Qubit

A classical bit is binary and deterministic. It exists strictly in state $|0\rangle$ or $|1\rangle$. A qubit is actually a state in space we call as Hilbert Space: a combination of $|0\rangle$ and $|1\rangle$, say

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

where $|\alpha|^2 + |\beta|^2 = 1$. These are not just values, but vectors with complex amplitudes. This probabilistic and phase-encoded nature of qubits leads to interference effects and parallelism in computation. While classical computation scales linearly with bits, quantum state space scales exponentially with qubits, 2^n .

Bits represent data using transistors. It enables control of voltage, which gives rise to classical bits — binary states that are either on or off. Qubits, on the other hand, require the manipulation of quantum states. It's not just about switching voltages; it's about controlling wavefunctions, preserving superposition, and maintaining coherence against environmental noise.

As for classical, we know:

- **Classical:** n bits $\rightarrow 2^n$ configurations (one at a time)
- **Quantum:** n qubits \rightarrow superposition of 2^n states simultaneously[18]

Table 1.1: Comparison between Classical Bits and Quantum Qubits

Feature	Classical Bit	Quantum Qubit
States	0 or 1	Superposition of 0 and 1
Physical Realization	Transistors	Superconducting loops, ions, etc
Measurement Outcome	Deterministic	Probabilistic
Parallelism	No	Yes (via superposition)
Entanglement	Not possible	Key resource
Behavior	Well-defined	Described by wavefunctions

1.3 Quantum Hardware vs Software

Classical computers run on bits (0 or 1), processed by logic gates built using transistors. Quantum computers use qubits, which need quantum hardware (e.g., superconducting circuits, trapped ions, photons). You cannot simulate all quantum problems effectively on classical computers due to exponential scaling in memory and time. Software written for quantum computers must match the hardware’s quantum nature — hence quantum circuits and quantum gates (not AND/NAND). “Every problem requires different hardware configurations” — quantum problems need tailored Hamiltonians and gate models.

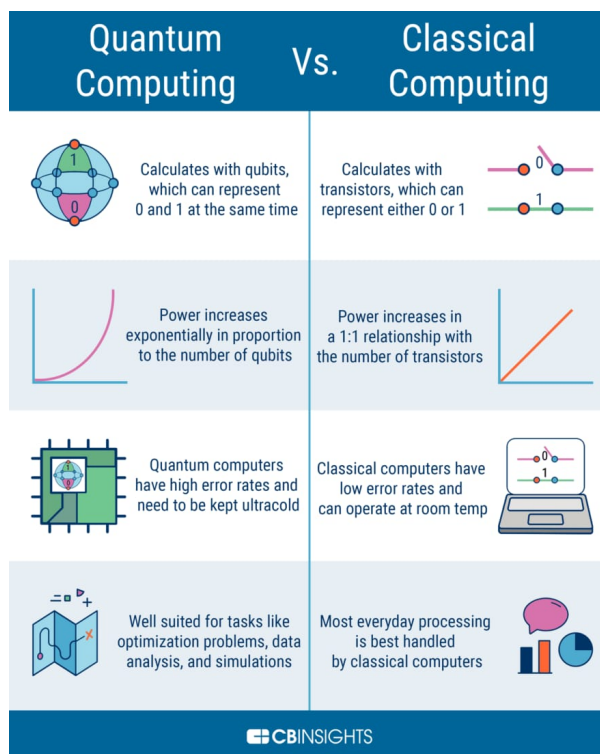


Figure 1.1: Comparing classical and quantum computing

When we study software, we often discuss processors. Similarly, in quantum computing, the nature of a quantum processor depends on the hardware platform being used. In superconducting qubits, Josephson junctions are controlled via microwave pulses. In ion trap systems, ions are suspended and manipulated using laser beams. Photonic quantum computers, on the other hand, use the polarization states of photons to represent qubits

Quantum effects like superposition, entanglement, and quantum interference can't be physically mimicked by classical transistors. You need qubit-specific manipulation: laser pulses, microwave gates, ion traps. Classical hardware might simulate small systems, but becomes intractable as number of qubits grows (due to exponential state space: 2^n states for n qubits).

$$10 \text{ qubits} \rightarrow 2^{10} = 1024 \text{ states}$$

$$100 \text{ qubits} \rightarrow 2^{100} \text{ states (too big for classical RAM)}[17]$$

1.4 Fundamental Concepts

In classical computing, each bit holds a single value at a time. In quantum computing, the state is not a single value until you measure it. In quantum physics, an eigenstate is the state you get after measurement. A quantum state is actually the superposition of all possible states before measurement. You write a quantum state like:

$$|\psi\rangle = a|000\rangle + b|111\rangle + c|101\rangle + \dots$$

The system evolves as a wavefunction, and you get one result upon measurement

This property of existing in multiple states at once is called superposition, and it allows quantum computers to have a large number of possibilities at a same time. Additionally, quantum states can become entangled, meaning the state of one qubit is directly linked to the state of other, irrespective of the fact that they may be very much far apart. These two principles — superposition and entanglement — are fundamental to quantum computation and enable the parallelism and computational advantages it offers over classical systems.[\[22\]](#)

1.5 Bloch Sphere Representation

The Bloch sphere is a geometrical representation of a qubit's state. Any single-qubit pure state can be mapped onto a point on the surface of the unit sphere:

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\phi}\sin\left(\frac{\theta}{2}\right)|1\rangle \quad (1.1)$$

Here, $\theta \in [0, \pi]$ and $\phi \in [0, 2\pi]$. The Bloch sphere visually encodes the qubit's phase and amplitude and helps illustrate quantum gate operations as rotations.

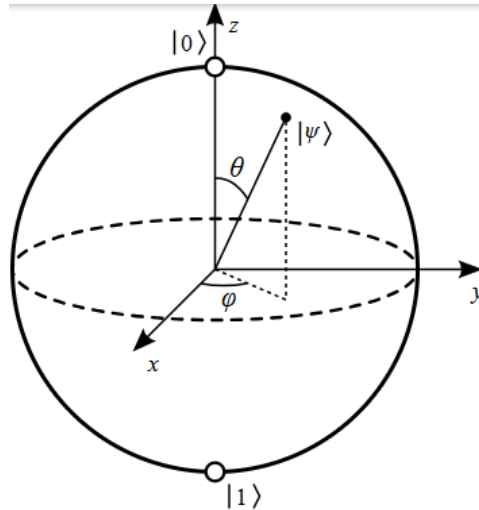


Figure 1.2: Bloch sphere, a geometrical representation of a two-level quantum system.

[24]

Quantum Gates as Bloch Sphere Rotations

Single-qubit gates can be interpreted geometrically as rotations of the state vector on the Bloch sphere. This visible illustration facilitates understanding of how quantum operations manipulate qubit states in a three-dimensional space.

Pauli-X Gate

The X gate acts as a bit-flip, rotating the qubit state by π radians around the X-axis. It maps $|0\rangle$ to $|1\rangle$ and vice versa.

The matrix form of the Pauli-X gate is:

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

Pauli-Y Gate

The Y gate performs a π -radian rotation around the Y-axis. It introduces both amplitude and phase change and is commonly used in entanglement and phase manipulation.

The matrix form of the Pauli-Y gate is:

$$Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$$

Pauli-Z Gate

The Z gate causes a π -radian rotation about the Z -axis. This operation flips the phase of the $|1\rangle$ component, leaving $|0\rangle$ unchanged. It is known as a phase-flip gate.

The matrix form of the Pauli-Z gate is:

$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

Hadamard Gate

The Hadamard (H) gate maps computational basis states to equal superpositions. Geometrically, it rotates the Bloch vector such that $|0\rangle$ becomes $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, and $|1\rangle$ becomes $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. On the Bloch sphere, this corresponds to a rotation from the poles to points on the equator.

The matrix form of the Hadamard gate is:

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

Rotation Gates

Parameterized rotation gates rotate the qubit around a specified axis by an arbitrary angle θ . These are fundamental in constructing arbitrary single-qubit unitaries:

- $R_x(\theta)$: Rotates the state vector by angle θ about the X -axis.
- $R_y(\theta)$: Rotates the state vector by angle θ about the Y -axis.

- $R_z(\theta)$: Rotates the state vector by angle θ about the Z-axis.

The matrix forms of these rotation gates are:

$$R_x(\theta) = \begin{pmatrix} \cos \frac{\theta}{2} & -i \sin \frac{\theta}{2} \\ -i \sin \frac{\theta}{2} & \cos \frac{\theta}{2} \end{pmatrix}$$

$$R_y(\theta) = \begin{pmatrix} \cos \frac{\theta}{2} & -\sin \frac{\theta}{2} \\ \sin \frac{\theta}{2} & \cos \frac{\theta}{2} \end{pmatrix}$$

$$R_z(\theta) = \begin{pmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{pmatrix}$$

Euler Decomposition of Arbitrary Gates

Any single-qubit unitary operation can be decomposed into a sequence of rotations using Euler angles. This typically takes the form:

$$U = e^{i\alpha} R_z(\beta) R_y(\gamma) R_z(\delta)$$

where the angles $\alpha, \beta, \gamma, \delta$ uniquely define the transformation.

Understanding quantum gates as rotations on the Bloch sphere enables better visualization of quantum logic. It also aids in optimizing circuits, correcting for noise, and designing intuitive gate sequences in hardware-aware implementations.[\[27\]](#)

1.6 Quantum Computing Architecture

Quantum computing architecture refers to the layered structure that integrates both the physical hardware responsible for manipulating qubits and the software that

controls, programs, and optimizes quantum operations. This architecture typically consists of three main layers: the quantum hardware, a control and interface layer, and the software stack. Together, these layers enable the development, simulation, and execution of quantum algorithms on different hardware platforms.

Quantum computing hardware varies across platforms such as:

1. Superconducting qubits (IBM, Google)
2. Trapped ions (IonQ, Honeywell)
3. Photonic qubits (PsiQuantum)
4. Topological qubits (Microsoft)

Each platform addresses the challenges of coherence, error correction, and gate fidelity differently.

Software stacks support the development and execution of quantum algorithms. Frameworks like Qiskit (IBM), Cirq (Google), PennyLane (Xanadu), and Classiq enable quantum programming, circuit design, and simulation.^[4]

1.7 Quantum Algorithms & Use-Cases

Quantum algorithms form the operational foundation of quantum computing, designed to solve specific problems more efficiently than classical counterparts by leveraging quantum phenomena such as superposition, entanglement, and interference. The following is a categorized overview of key quantum algorithms relevant to theoretical exploration and practical simulation tasks.

1. Deutsch–Jozsa Algorithm: One of the first quantum algorithms, developed to demonstrate a clear advantage over classical approaches. It distinguishes between constant and balanced Boolean functions using only a single evaluation. While its

direct applications are limited, it helped establish the framework for future algorithm development.

2. **Grover's Algorithm:** This search algorithm provides a quadratic speed-up for unstructured search problems, reducing the time complexity from $O(N)$ to $O(\sqrt{N})$. It uses amplitude amplification techniques to increase the probability of the correct result upon measurement. Grover's algorithm is broadly applicable to optimization and decision-making tasks.
3. **Shor's Algorithm:** A breakthrough in quantum computing, Shor's algorithm factors large integers exponentially faster than the best-known classical algorithms. It utilizes modular exponentiation and the quantum Fourier transform to extract periodicity. Its success underscores the potential of quantum computing to break classical cryptographic systems like RSA.
4. **Quantum Phase Estimation (QPE):** A foundational subroutine used to estimate the eigenvalues of a unitary operator given an eigenstate. It is critical in applications such as finding energy eigenvalues in quantum chemistry and physics. QPE works by encoding phase information in ancillary qubits and extracting it via inverse quantum Fourier transform.
5. **Trotter-Suzuki Decomposition:** Used in quantum simulation to approximate the exponential of a sum of non-commuting operators. This method enables simulation of time evolution under complex Hamiltonians by breaking them into simpler, sequential unitary steps, making it vital for implementing models like Ising and Heisenberg on quantum circuits.
6. **Variational Quantum Eigensolver (VQE):** A hybrid algorithm combining quantum state preparation with classical optimization. It approximates ground-state energies of Hamiltonians using parametrized quantum circuits. VQE is particularly effective on Noisy Intermediate-Scale Quantum (NISQ) hardware due to its reduced quantum resource requirements.
7. **Quantum Approximate Optimization Algorithm (QAOA):** Designed for solving combinatorial optimization problems. QAOA constructs an ansatz using alter-

nating layers of unitary operators derived from a cost Hamiltonian and a mixer Hamiltonian. Classical optimization tunes the parameters to find approximate solutions to problems like Max-Cut or portfolio optimization.

8. Quantum Machine Learning (QML) Algorithms: These include various models that integrate quantum circuits into machine learning workflows. Key approaches are:

- Quantum kernel methods: Embed data into high-dimensional Hilbert space using quantum feature maps.
- Variational classifiers: Use trainable quantum circuits for supervised tasks.
- Quantum generative models: Generate probability distributions using quantum interference, potentially useful for unsupervised learning.

QML holds promise for accelerating pattern recognition and data analysis tasks under certain conditions.

Uses:

Quantum computing is especially suited for solving certain types of problems more efficiently than classical computers. Some key application areas include Drug discovery, Climate modeling, Cryptography, Optimization and Agriculture. Not all classical problems need quantum solutions. Quantum computing excels only where quantum mechanical properties (superposition, entanglement and interference) offer real advantages.[\[16\]](#)

Table 1.2: Important Quantum Algorithms Overview

Quantum Fourier Transform	Quantum Phase Estimation	Shor Algorithm
Performs a Fourier transform on qubits; essential in many quantum algorithms.	Estimates the eigenvalue (phase) of a unitary operator; key in quantum chemistry and factoring.	Efficiently factors large integers; breaks RSA encryption.
Grover Algorithm	HHL Algorithm	Quantum Support Vector Machine
Searches an unsorted database in \sqrt{N} time; faster than classical $O(N)$.	Solves systems of linear equations exponentially faster than classical methods.	Applies SVM on quantum hardware; used in classification tasks.
Quantum k-means Clustering	Quantum Principal Component Analysis	Quantum Q Learning
Performs clustering using quantum distance evaluation and superposition.	Identifies principal components of quantum data efficiently.	Quantum-enhanced reinforcement learning using qubits and quantum memory.
Quantum Approximate Optimization Algorithm	Variational Quantum Eigensolver	
Solves combinatorial optimization problems using a hybrid quantum-classical approach.	Finds the ground state energy of a system; widely used in quantum chemistry.	

1.8 Quantum Computing Roadmaps

Global roadmaps project different timelines for scalable, fault-tolerant quantum computing:

1. IBM: 1,000+ qubit chips (Condor) by 2025 and modular quantum systems beyond 2026.
2. Google: Achieving "quantum supremacy" with > 50 qubits; targeting fault-tolerant computing in a decade.
3. China: Jiuzhang and Zuchongzhi supercomputers pushing boson sampling.

The quantum computing landscape is rapidly evolving, with several tech giants and startups outlining ambitious development roadmaps. This section summarizes the strategic plans and projections of key players in the field, including IBM, Google, Microsoft, Rigetti, Pasqal, D-Wave, IonQ, and Quantinuum.

IBM

IBM's roadmap, extending through 2033, envisions the deployment of a quantum-centric supercomputer by 2025, integrating over 4,000 qubits. A major focus lies on increasing circuit depth and quality, aiming to execute over 5,000 gates with parametric circuits. The company's architecture emphasizes modularity, with the IBM Quantum System Two expected to accommodate up to 16,632 qubits. The Heron processor marks a critical milestone in their pursuit of quantum error correction and autonomous middleware optimization. According to IBM leadership, near-term quantum applications will primarily benefit materials science and chemistry.

Google

Google's long-term quantum initiative, first introduced in 2020, targets a fault-tolerant, utility-grade quantum computer by 2029. Following the milestone of quantum supremacy

in 2019 using the 53-qubit Sycamore processor, they have shifted focus to logical qubit construction and error reduction techniques. The Willow chip and a functional prototype of a logical qubit demonstrate tangible progress. With substantial investments and infrastructure centered at their Santa Barbara facility, Google aims to tackle global challenges, including drug discovery and climate modeling, within the next decade.

Microsoft

Microsoft’s strategy is centered on achieving scalability through topological qubits. Their roadmap is divided into three phases—Foundational, Resilient, and Scale—culminating in utility-scale quantum computing. In early 2025, the company introduced the Majorana 1 processor, which is designed for million-qubit scalability using hardware-encoded fault tolerance. A key scientific breakthrough was the observation of Majorana zero modes in 2023. Microsoft’s involvement in DARPA’s US2QC program further underlines their commitment to constructing a fault-tolerant system within a few years.

Rigetti Computing

Rigetti aims to scale from its current 84-qubit Ankaa-3 platform to a 336-qubit system named Lyra. Their roadmap, last updated in Q3 2024, targets over 100 qubits by the end of 2025. The architecture is based on superconducting qubits with multi-chip integration, supported by a proprietary technique called Alternating-Bias Assisted Annealing (ABAA). With two-qubit gate fidelity exceeding 99%, the company is positioning itself as a leader in quantum machine learning and optimization use cases.[\[25\]](#)

Table 1.3: Comparative Overview of Quantum Computing Roadmaps and Milestones

Company	Key Milestone	Timeline
IBM	Quantum-centric supercomputer	2025
Google	Error-corrected quantum computer	2029
Microsoft	Million-qubit chip with topological qubits	Newly introduced
Rigetti	Exceeding 100 qubits	End of 2025

1.9 Error and Decoherence in Quantum systems

These issues form one of the *two major problem types* in quantum computing During hardware construction which includes Physical noise sources, decoherence time, gate fidelity, and connectivity limitations. And after hardware is built, it includes Algorithmic errors, software-layer limitations, circuit depth, and measurement complexity. Quantum computing faces significant hardware-level challenges that directly impact simulation accuracy and scalability. Among the most critical issues are:

1. Noise: Uncontrolled interactions with the environment introduce errors into quantum computations.
2. Decoherence: Qubits lose their quantum properties (superposition and entanglement) over time due to interactions with external systems.
3. Gate Errors: Imperfections in applying quantum gates lead to deviations from ideal unitary operations.
4. fError Mitigation: A class of techniques (discussed in lecture notes) aimed at reducing the impact of errors without full-fledged quantum error correction.

Understanding and mitigating these errors is crucial, especially for accurate quantum Hamiltonian simulation, where even small errors can propagate and distort the system's time evolution.[\[18\]](#)

Chapter 2

Quantum Computing Literature

2.1 Quantum Gates Representation

Quantum gates are the basic operations that change the state of qubits in a quantum computer. Just like classical computers use logic gates (like AND, OR, and NOT) to process bits, quantum computers use quantum gates to control qubits, which can exist in combinations of 0 and 1 at the same time because of a property called superposition.

Quantum gates are the building blocks of quantum computation, serving as unitary operators that manipulate qubit states in a reversible manner. Unlike classical logic gates which operate on bits through fixed boolean functions, quantum gates preserve quantum coherence and operate through transformations on the Hilbert space of states.

Each quantum gate is a precise operation that adjusts the probability amplitudes of a qubit's state or entangles multiple qubits together. These gates are usually represented by matrices, and when applied to qubits, they transform the qubit states through matrix multiplication. Unlike irreversible classical logic, all quantum gates are reversible and preserve probability.

Quantum gates manipulate qubit states via unitary operations. Basic gates include:

1. Pauli X gate: Flips qubit.
2. H (Hadamard): Introduces superposition.
3. T and S: Phase gates.
4. CNOT, Toffoli: Entangle qubits and build universal logic.

Quantum Gate Representation

The Hadamard gate on qubit 0 creates a superposition $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, and the CNOT gate entangles the two qubits, resulting in the maximally entangled Bell state. This demonstrates how unitary quantum gates can generate non-classical correlations fundamental to quantum computing.^[17]

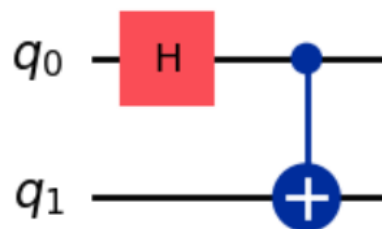


Figure 2.1: Bell State

We show the implementation of the code for the calculation of Bell States in the appendix.(A)

2.1.1 Types of Quantum Gates

Single-Qubit Gates:

Operations on individual qubits correspond to unitary 2×2 matrices. These gates can be visualized as rotations of the qubit vector on the Bloch sphere. A general single-qubit gate U satisfies the condition $U^\dagger U = I$, ensuring probability conservation.

Fundamental Gates:

The Pauli gates (X , Y , and Z) represent π -radian rotations about the respective axes of the Bloch sphere. The identity gate I leaves the qubit unchanged. The Hadamard gate H generates superposition, acting as a rotation between computational and diagonal bases.

Parameterized Rotation Gates:

Gates such as $R_x(\theta)$, $R_y(\theta)$, $R_z(\theta)$ allow continuous control of the qubit state by applying a rotation of angle θ about a specified axis. These gates are crucial for tuning variational circuits and implementing precise state evolution.

Multi-Qubit Gates:

To generate entanglement and perform conditional operations, two-qubit gates are required. The Controlled-NOT (CNOT) gate is a key example, flipping the target qubit based on the control qubit's state. Its matrix form is:

$$\text{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

This gate cannot be decomposed into single-qubit operations and is essential for creating entangled Bell states.

Universal Gates:

A universal gate set enables construction of any unitary operation to arbitrary precision. A standard universal set includes the Hadamard gate, phase gate S , $\pi/8$ gate T , and the CNOT gate. This minimal set is sufficient to approximate any quantum algorithm.[17]

2.1.2 Physical Relevance:

Quantum gates must be implemented within the constraints of hardware platforms. For example, superconducting qubits often natively support R_x and CNOT gates, while trapped ions favor Mølmer–Sørensen interactions. Therefore, gate synthesis and transpilation are tailored to the device’s native operations.

Quantum gates provide the abstract framework and physical mechanisms by which quantum algorithms are realized. Their unitary nature ensures reversibility, and their geometrical interpretation on the Bloch sphere enables visual insight into quantum dynamics. Mastery of gate operations is foundational to quantum circuit design and algorithm implementation.[18]

2.1.3 Unitary Transformations

All quantum evolutions are governed by unitary matrices U satisfying the condition $U^\dagger U = I$, where U^\dagger is the Hermitian conjugate of U , and I is the identity matrix. This unitary nature guarantees conservation of probability (norm preservation) and the reversibility of time evolution. In practice, quantum circuits implement these evolutions as ordered sequences of unitary gates acting on initial quantum states, resulting in final states that can be measured to extract computational outcomes.

1. **Probability Preservation:** In quantum mechanics, the squared magnitude of the wavefunction's amplitude corresponds to measurable probabilities. Unitary operators preserve the inner product of states, ensuring that the total probability remains conserved during any evolution: $\langle\psi|\psi\rangle = \langle U\psi|U\psi\rangle$.
2. **Time Evolution and Schrödinger Equation:** The evolution of an isolated quantum system is governed by a unitary operator derived from the time-dependent Schrödinger equation. For a time-independent Hamiltonian H , the time-evolution operator is given by:

$$U(t) = e^{-iHt/\hbar}$$

This exponential of a Hermitian operator yields a unitary operator, highlighting the fundamental role of unitary evolution in the dynamics of quantum systems.

3. **Gate Representation:** In quantum computing, all valid quantum gates are represented by unitary matrices. These gates manipulate qubit states without loss of quantum information. For instance, the Pauli matrices, Hadamard gate, phase gate, and controlled operations all satisfy unitarity and therefore correspond to physically realizable transformations.
4. **Geometric Interpretation:** For a single qubit, unitary transformations correspond to rotations on the Bloch sphere. A general unitary 2×2 matrix (up to global phase) can be decomposed using Euler angles as:

$$U = e^{i\alpha} R_z(\beta) R_y(\gamma) R_z(\delta)$$

This decomposition highlights the continuous control available in single-qubit operations through rotational degrees of freedom.

5. **Reversibility and Quantum Rewinding:** Every unitary transformation is invertible, with its inverse given by the conjugate transpose: $U^{-1} = U^\dagger$. This reversibility is a critical property in quantum algorithms, allowing computation paths to be reversed or uncomputed—essential for techniques like phase estimation or Grover's diffusion operator.
6. **Universality and Composition:** Any complex unitary operation can be approximated to arbitrary precision by a finite sequence of elementary unitary gates from

a universal set. This includes decomposing high-dimensional unitary matrices into tensor products and controlled operations across multiple qubits.

7. **Role in Quantum Simulations:** Key quantum algorithms such as the Quantum Fourier Transform (QFT), phase estimation, and variational algorithms rely heavily on structured unitary transformations. We use these algorithms to simulate efficient unitary circuits that approximate desired transformations with minimal overhead remains an active area of research.

Unitary transformations are not merely a formal requirement—they encode the fundamental reversibility and coherence of quantum processes. Their algebraic properties and geometric interpretations bridge abstract quantum theory with implementable computational logic, making them indispensable tools in the architecture of quantum algorithms.[\[17\]](#)

2.2 Fields in Quantum Computing

1. Quantum Simulation

Quantum simulation is one of the most promising and impactful areas within quantum computing. It involves using a quantum computer to model the behavior of complex quantum systems that are difficult to study with classical computers. Applications include:

1. Predicting molecular structures and chemical reactions for drug discovery.
2. Designing new materials and superconductors.
3. Studying particle interactions and exotic states in high-energy physics.

Quantum systems naturally obey quantum rules, making quantum computers the most suitable tools to simulate such systems efficiently and accurately. Quantum simulation is expected to revolutionize fields like chemistry, material science, and energy technology.[\[9\]](#)

2. Quantum Cryptography

Quantum cryptography focuses on developing secure communication methods based on the laws of quantum mechanics. The most notable technique is **Quantum Key Distribution (QKD)**, which ensures unbreakable encryption by detecting any eavesdropping attempts in real-time. It is the classical counterpart of RSA which is responsible for secure communication between classical computers.[\[10\]](#)

3. Quantum Machine Learning (QML)

This field merges quantum computing with artificial intelligence. It involves creating algorithms capable of processing and analyzing large data sets more efficiently by utilizing quantum parallelism and entanglement. Applications include pattern recognition, optimization, and financial modeling.[\[1\]](#)

2.3 Analytical vs Numerical (Classical) Simulations

Simulating physical systems using their Hamiltonians is a cornerstone application of quantum computing. Various methods exist depending on the complexity and solvability of the system.

Analytical methods involve solving equations directly using classical computing when closed-form solutions exist. These methods are often limited to small or idealized systems, and are well-covered in foundational quantum mechanics texts.

Numerical simulations approximate solutions when analytical forms are intractable. Classical approaches involve discretization, iterative solvers, and matrix diagonalization, but they suffer from exponential scaling with system size.

Hamiltonian simulation is a quantum algorithmic approach used to model time evolution of quantum systems by simulating the unitary operator e^{-iHt} , where H is the Hamiltonian of the system. A key method involves breaking H into a sum of simpler

terms and applying a Trotter-Suzuki decomposition:

$$e^{-iHt} \approx \left(e^{-iH_1 t/n} e^{-iH_2 t/n} \dots e^{-iH_k t/n} \right)^n$$

This technique enables efficient simulation of local interactions in many-body systems.

Hybrid algorithms, such as the Variational Quantum Eigensolver (VQE), combine quantum circuits and classical optimization to approximate ground states of Hamiltonians. VQE is particularly useful for molecular and material simulations where full quantum evolution is too costly.

Quantum simulation is relevant to fields including chemistry, condensed matter physics, and lattice gauge theories. However, classical simulation of Hamiltonian dynamics becomes exponentially difficult with increasing system size, motivating the need for quantum approaches.

Limitation: Classical simulation of Hamiltonian dynamics scales poorly—memory and computation time grow exponentially with system size.[\[9\]](#)

Table 2.1: "Types of Simulation Methods" A diverse approach used to model, analyze and predict the behaviour of complex systems

Monte Carlo Simulation	Agent-Based Simulation	Discrete-Event Simulation	Continuous Simulation
Uses random sampling to estimate outcomes, often for risk assessment.	Models interactions of autonomous agents to observe overall system effects.	Models systems as sequences of events to optimize processes.	Simulates systems with continuous changes, using differential equations.
Hybrid Simulation	System Dynamics	Hamiltonian Simulation	Quantum Hamiltonian Simulation
Combines discrete-event and continuous simulations for complex systems.	Focuses on feedback loops in systems for long-term behavior analysis.	Simulates physical systems using Hamiltonian mechanics and energy principles.	Models quantum systems by evolving them under quantum Hamiltonians.

2.4 Motivation for Quantum Hamiltonian Simulations

Classical simulations of quantum systems scale poorly due to exponential state spaces. Quantum computers can simulate dynamics more naturally, as proposed by Feynman. Applications include:

- Studying material phases (Ising/Heisenberg models)
- Predicting chemical reactions
- Exploring non-equilibrium quantum dynamics

Quantum simulation holds promise to revolutionize research in chemistry, materials science, and fundamental physics. Quantum hardware excels in simulating quantum systems by directly emulating Hamiltonian evolution—avoiding the exponential blowup faced in classical simulations.

Key quantum algorithms used for Hamiltonian simulation include:

- **Quantum Phase Estimation (QPE)**
- **Variational Quantum Eigensolver (VQE)**
- **Trotterized time evolution**

These algorithms are instrumental in simulating:

- Chemical reactions at the quantum level
- Material properties in condensed matter physics
- Complex quantum dynamics in strongly correlated systems

By leveraging quantum parallelism and unitary evolution, these methods provide efficient pathways to understanding systems that are otherwise intractable using classical computation.[\[8\]](#)

2.5 Shifting to Pauli Matrices

Quantum observables and gate generators often use the Pauli matrices:

$$\sigma_x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad \sigma_y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \quad \sigma_z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}.$$

These matrices form the basis of the Lie algebra $\mathfrak{su}(2)$, which is fundamental to quantum operations such as rotations and time evolution. In quantum computing, they serve as building blocks for more complex gate constructions and Hamiltonians.[\[17\]](#)

2.6 Building Quantum Circuits

Quantum circuits are constructed by applying a series of quantum gates that manipulate qubits according to an algorithm's logical structure. Each gate corresponds to a unitary operation acting on one or more qubits, and the entire circuit represents a sequence of such operations.

The modularity of quantum circuit design allows reuse of subroutines and facilitates optimizations. Complex algorithms, such as Grover's search or Shor's factoring, are built from fundamental gate operations like the Hadamard, CNOT, and phase gates.

Circuit implementation on actual quantum hardware requires *transpilation*—the process of converting high-level circuits into gate sequences compatible with the hardware's native gate set and connectivity constraints. This step is essential for optimizing fidelity, minimizing decoherence, and reducing circuit depth.

Quantum circuit design is a central task in quantum algorithm development, providing the blueprint for how quantum information is manipulated throughout a computation.[\[17\]](#)

2.7 Methods and Technologies for Quantum Information Processing

Several technologies are explored to encode and process qubits:

- **Photonic Qubits:** Fast and low-loss communication (Kok & Loveett).

- **Trapped Ions:** High-fidelity gates and long coherence.
- **Superconducting Circuits:** Scalable integration.
- **Topological Qubits:** Noise-resistant by design.
- **Spin Qubits:** Use electron/nuclear spins in quantum dots.

Each has distinct advantages and research challenges in fabrication, control, and integration.[\[14\]](#)

2.8 Simulation Workflow

The process of simulating physical systems differs significantly between classical and quantum approaches:

Classical Workflow

- Define the Hamiltonian of the system
- Build a numerical model (e.g., discretization, basis selection)
- Approximate the solution using algorithms (e.g., diagonalization, time-stepping)
- Simulate system dynamics or properties
- Analyze results (e.g., energies, correlations)

Quantum Workflow

- Map the Hamiltonian to qubit operators (e.g., using Pauli matrices)
- Implement the corresponding quantum circuit
- Prepare an appropriate initial quantum state

- Execute quantum gates to simulate evolution or interaction
- Perform measurement on qubits
- Extract results such as probabilities and expectation values[\[17\]](#)

Chapter 3

Programming with Qiskit and Jupyter Notebook

3.1 Overview

Qiskit, the open-source quantum SDK developed by IBM, is widely used for writing, simulating, and running quantum algorithms. This chapter outlines the environment setup used in this thesis, versions of Qiskit applied, package installations, encountered issues, and resolution strategies through virtual environments.

3.2 Development Environment Setup

3.2.1 Conda Environment Using Miniconda

To ensure version control and compatibility, multiple Python environments were created using Miniconda and Anaconda Prompt. For clean separation of Qiskit versions and dependencies, the following conda environment named **CWQ** was created:

- **Step 1:** `conda create --name CWQ`

- **Step 2:** Confirm installation with Y
- **Step 3:** `conda activate CWQ`
- **Step 4:** `conda install pip`
- **Step 5:** Confirm installation with Y
- **Step 6:** `pip install qiskit`
- **Step 7:** `pip install matplotlib`
- **Step 8:** `pip install qiskit_ibm_runtime`
- **Step 9:** `pip install pylatexenc`
- **Step 10:** `pip install qiskit_aer`

3.2.2 Packages Installed in System Command Prompt

Another set of packages were installed outside the conda environment via the Command Prompt to ensure compatibility with Jupyter Notebook and latest Qiskit versions:

- `pip install notebook`
- `pip install matplotlib`
- `pip install qiskit`
- `pip install qiskit==2.0.0 qiskit-aer==0.17`
- `python -m venv myenv`
- `pip install qiskit`
- `jupyter notebook`
- `pip install qiskit-ibm-provider`
- `pip install qiskit-algorithms (version 0.3.1)`
- Check Qiskit packages using: `pip list | findstr qiskit`[\[21\]](#)

3.3 Qiskit Versioning and Practical Considerations

Initially, Qiskit version 3.13 was tested but found incompatible with key features needed for this thesis. Hence, Qiskit versions 3.11 and 3.10 were used during most simulations. For compatibility with legacy code (e.g., using COBYLA optimizer (Qiskit-Constrained Optimization BY Linear Approximation (COBYLA) algorithm) is a numerical optimization method for constrained problems where the derivative of the objective function is not known.) and VQE algorithm), two isolated environments were created:

- **CWQ:** Qiskit 2.0.0 — used for simulations with new primitives like `BaseSamplerV2`.
- **DOWNGRADE:** Qiskit 1.4.0 — used when older code (e.g., VQE requiring `BaseSampler`) produced errors in 2.x versions.

3.4 Problems Encountered and Mitigation Strategy

During execution of VQE algorithms using the COBYLA optimizer within the CWQ environment, errors emerged due to deprecated primitives. Specifically:

- VQE and COBYLA relied on the original `BaseSampler`, which is no longer supported in Qiskit 2.0.0 and above.
- The updated primitive `BaseSamplerV2` was introduced in Qiskit 2.0.0, rendering older code incompatible.
- Rather than refactoring or replacing legacy code, a more practical approach was implemented: a new environment called **DOWNGRADE** was created and configured with Qiskit 1.4.0.

This modular setup allowed for seamless switching between environments depending on the code requirements, preventing unnecessary debugging or refactoring.

3.5 Qiskit Primitives Used

3.5.1 Estimator Primitive

The `Estimator` class computes expectation values of observables on quantum states. It integrates with Qiskit Runtime and supports both real device and simulator backends.

3.5.2 Sampler Primitive

The `Sampler` returns shot-by-shot bitstrings sampled from the final quantum state. It is used for probabilistic simulations and circuit sampling analysis.

3.5.3 Related Imports

Example imports used frequently:

Listing 3.1: Qiskit Primitive Imports

```
from qiskit_ibm_runtime import Sampler
from qiskit import transpile
from qiskit_aer import Aer
```

3.6 Unitary Gate Utility

The `UnitaryGate` in Qiskit allows users to define arbitrary quantum gates represented by unitary matrices. This is essential when implementing non-standard operations such as \sqrt{Y} or rotation gates not included in the basic library.

3.7 IBM Cloud Access History

IBM introduced cloud-based quantum computing on May 4, 2016. As of today, IBM provides a series of quantum access plans enabling users to execute circuits on real quantum processors.^[20]

3.8 Qiskit Circuit Transpilation and Measurement Bits

3.8.1 Declaring Classical Bits in Quantum Circuits

In Qiskit, it is important to distinguish between quantum and classical bits when building circuits. The use of `num_cl_bits` (classical bits) instead of `num_qubits` becomes essential in situations involving measurement or classical control logic.

- Quantum measurements project quantum states onto classical bits.
- If your circuit only simulates quantum evolution without any measurements or classical feedback, declaring classical bits might be unnecessary.
- In scenarios involving **measurement**, **reset**, or **classical-controlled operations**, it is mandatory to explicitly allocate classical bits in the circuit.

3.8.2 Transpilation and Logical to Physical Qubit Mapping

When a quantum circuit is **transpiled**, Qiskit optimizes it to fit the topology of the target quantum device. During this process, logical qubits from the circuit may be mapped to physical qubits on the actual hardware.

This qubit remapping is crucial for ensuring that the observables measured correspond to the correctly mapped qubits in the transpiled circuit.

3.8.3 Comparison of SamplerV2 and EstimatorV2 Primitives

Qiskit's runtime includes advanced primitives such as **SamplerV2** and **EstimatorV2**, each optimized for different use-cases in quantum computation.[\[19\]](#) Their functionality is compared in the table below:

Table 3.1: Comparison between SamplerV2 and EstimatorV2 Primitives

Feature	SamplerV2	EstimatorV2
Returns	Probability distribution over outcomes (bitstrings)	Expectation values of observables
Input	Quantum circuits only	Circuits and observables (e.g., H)
Use Case	Sampling (e.g., for measurements)	Computing physical quantities
Typical Output	$\{ \text{'000': 0.25, '111': 0.75} \}$	e.g., $\langle \psi H \psi \rangle$

Chapter 4

Spin Systems

4.1 Spin Systems in Quantum Mechanics and Quantum Computing

Spin systems are pivotal in the study of quantum many-body physics and are essential in understanding the behavior of complex quantum materials. Moreover, they offer a testbed for evaluating quantum computational methods, especially in simulating physical systems that are intractable for classical computation. This chapter presents an in-depth, original overview of spin systems, referencing foundational texts such as Nielsen and Chuang's *Quantum Computation and Quantum Information* and recent studies in quantum machine learning applied to Ising and Heisenberg models.

4.1.1 Quantum Spin: Fundamental Concepts

Quantum spin is a non-classical property, inherent to particles, without any classical analog of actual rotation. The most fundamental representation is the spin- $\frac{1}{2}$ particle, whose state resides in a two-dimensional Hilbert space. The basis states are:

$$|\uparrow\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad |\downarrow\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

These are eigenstates of the Pauli- Z matrix:

$$\sigma_z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

In combination with the Pauli- X and Pauli- Y operators:

$$\sigma_x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad \sigma_y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$$

they obey the $SU(2)$ algebra:

$$[\sigma_i, \sigma_j] = 2i\epsilon_{ijk}\sigma_k$$

where ϵ_{ijk} is the Levi-Civita symbol.[\[17\]](#)

4.2 Spin Lattice Models

In condensed matter and quantum computation, lattice-based spin systems model interacting qubits arranged in regular spatial structures. Each site of a lattice hosts a quantum spin, and the system's dynamics are governed by a Hamiltonian that encodes both spin-spin interactions and interactions with external fields.

4.2.1 The Ising Model

it's a mathematical model used in statistical mechanics to understand phase transitions, how a material goes from being magnetic to non-magnetic when heated. The classical Ising model consists of spins on a lattice that can be either up or down, and they interact with their nearest neighbors.

In quantum mechanics, The 1D transverse-field Ising model is defined by the Hamiltonian:

$$H = -J \sum_i \sigma_z^{(i)} \sigma_z^{(i+1)} - h \sum_i \sigma_x^{(i)}$$

where J is the interaction strength and h denotes the transverse magnetic field. This model is paradigmatic in exploring quantum phase transitions, which occur when h/J crosses a critical threshold. The TFIM is also used as a benchmark in quantum simulation algorithms and near-term quantum hardware implementations.

4.2.2 The Heisenberg Model

The Heisenberg Hamiltonian includes interactions in all spatial directions:

$$H = J \sum_i (\sigma_x^{(i)} \sigma_x^{(i+1)} + \sigma_y^{(i)} \sigma_y^{(i+1)} + \sigma_z^{(i)} \sigma_z^{(i+1)})$$

Special cases include the XX model (only x and y terms), XXZ model (anisotropy in z), and XYZ model (fully anisotropic). These models are central to the study of magnetic ordering, entanglement, and many-body localization and quantum chaos.[\[23\]](#)

4.3 Classical vs Quantum Ising Model

The Ising model is a cornerstone of statistical mechanics and quantum simulation, used to understand magnetism and critical phenomena, and phase transitions. The classical and quantum versions of the model differ significantly in their mathematical formulation, physical behavior, and computational complexity.

4.3.1 Classical Ising Model

The classical Ising model considers a lattice of binary spin variables $s_i \in \{-1, +1\}$ representing magnetic dipole moments, arranged on a one-, two-, or three-dimensional grid. The energy of a configuration is given by:

$$E = -J \sum_{\langle i,j \rangle} s_i s_j - h \sum_i s_i$$

Here, J denotes the interaction strength between neighboring spins $\langle i, j \rangle$, and h is the strength of an external magnetic field. The model is purely probabilistic and temperature-dependent, with the probability of a configuration governed by the Boltzmann distribution:

$$P(\{s_i\}) \propto e^{-E/k_B T}$$

where k_B is Boltzmann's constant and T is the temperature. Classical Ising models are used to study thermal phase transitions, such as the onset of spontaneous magnetization, and are typically solved using Monte Carlo methods, mean-field theory, and renormalization group methods.

4.3.2 Quantum Ising Model

In the quantum extension of the Ising model, the spin variables become quantum operators, and the model includes non-commuting terms that introduce quantum fluctuations. The 1D transverse-field quantum Ising model is expressed by the Hamiltonian:

$$H = -J \sum_i \sigma_z^{(i)} \sigma_z^{(i+1)} - h \sum_i \sigma_x^{(i)}$$

The Pauli operators $\sigma_z^{(i)}$ and $\sigma_x^{(i)}$ act on qubit i , where the z term captures classical interactions between neighboring spins, and the x term introduces a transverse field that drives transitions between spin states.

Unlike the classical model, the quantum version exhibits a **quantum phase transition** at zero temperature as the ratio h/J is varied. These transitions arise from changes in the ground state of the system due to quantum fluctuations, rather than thermal effects. The model is central to quantum simulation research and serves as a benchmark for studying criticality, entanglement scaling, and non-equilibrium dynamics on quantum computers.[\[6\]](#)

4.3.3 Key Differences

Aspect	Classical Ising Model	Quantum Ising Model
Spin Variables	$s_i \in \{-1, +1\}$	Pauli matrices (σ_z, σ_x)
State Space	Discrete configurations	Hilbert space superpositions
Interactions	Neighboring spins, static field	Includes quantum tunneling (transverse field)
Dynamics	Thermal fluctuations	Quantum fluctuations
Phase Transitions	Thermal (finite T)	Quantum (at $T = 0$)
Solvability	Analytical (1D), Approximate (2D/3D)	Numerical (Trotterization, Tensor Networks)

Table 4.1: Comparison of classical and quantum Ising models

4.3.4 Implications for Quantum Computing

The quantum Ising model can be efficiently mapped onto a quantum computer, making it a benchmark problem for quantum simulation. Its non-commuting Hamiltonian terms are ideal for exploring Suzuki-Trotter decomposition and variational techniques such as QAOA (Quantum Approximate Optimization Algorithm). By discretizing time evolution using Trotterization or employing variational ansätze, quantum devices can approximate the ground state or dynamics of the model. These methods allow researchers to explore quantum phase transitions, entanglement, and many-body localization directly on quantum hardware. In contrast, the classical Ising model serves as a reference for validating quantum simulations and understanding the limits of classical approximation.^[7]

4.4 Quantum Simulation Techniques

Quantum simulation maps these spin Hamiltonians onto the dynamics of qubits on a quantum processor. Time evolution under a Hamiltonian H can be approximated using the Trotter-Suzuki decomposition:

$$e^{-iHt} \approx \left(e^{-iH_A \Delta t} e^{-iH_B \Delta t} \right)^n$$

where $H = H_A + H_B$ with non-commuting components. This method allows modular construction of quantum circuits that mimic the physical evolution of Ising and Heisenberg systems.

Variational algorithms such as the Variational Quantum Eigensolver (VQE) and Quantum Approximate Optimization Algorithm (QAOA) are also employed to approximate ground states by optimizing parameterized quantum circuits against a cost function derived from energy expectations.^[9]

4.5 Machine Learning for Spin Models

Quantum and classical machine learning have been increasingly applied to characterize spin systems. Methods include:

- Supervised learning to detect quantum phase transitions.
- Reinforcement learning for quantum control of spin chains.
- Quantum neural networks for Hamiltonian learning.

Quantum classifiers are also capable of distinguishing between different quantum states based on entanglement and magnetization features learned from sample data.[\[3\]](#)

4.6 Experimental Realizations

Spin models are experimentally realized on diverse platforms:

- **Trapped Ions:** Emulate spin-spin coupling via phonon-mediated interactions.
- **Superconducting Circuits:** Implement controllable couplings in qubit lattices.
- **Optical Lattices:** Ultracold atoms simulate lattice Hamiltonians naturally.
- **NV Centers:** Defects in diamond used for coherent spin manipulation.

These platforms allow physical validation of theoretical models and benchmarking of quantum simulators.[\[9\]](#)

4.7 Transverse Field Ising Model and Custom Unitary Gates

4.7.1 1D Transverse Field Ising Model (TFIM)

The one-dimensional Transverse Field Ising Model (1D TFIM) is a foundational model in quantum simulations that captures the behavior of spin chains with nearest-neighbor interactions under an external transverse magnetic field. It is mathematically defined by the Hamiltonian:

$$H = -J \sum_{\langle i,j \rangle} Z_i Z_j - h \sum_i X_i$$

where:

- J represents the interaction strength between adjacent spins,
- $Z_i Z_j$ denotes the nearest-neighbor interaction (Ising term),
- h is the strength of the transverse field,
- X_i applies the transverse magnetic field on site i .

This model is widely used to benchmark quantum algorithms, especially in demonstrating primitives and quantum gate dynamics.

4.7.2 Custom Unitary Gates in Qiskit

Custom unitary gates are not pre-defined in Qiskit. Instead, they are created manually by the user using a specific unitary matrix to perform transformations not available in the standard gate library.

- **Built-in Gates:** Common gates include X , Y , H , CX , T , etc.

- **Custom Gates:** Implemented using `UnitaryGate()` with a defined unitary matrix — such as \sqrt{Y} , arbitrary rotations, or time evolution operators like e^{-iHt} .

Example Use Case: Suppose we want to apply \sqrt{Y} (the square root of the Pauli-Y gate). Since it's not natively available, we define it using a unitary matrix.

4.7.3 The Pauli-Y Gate

The Pauli-Y gate is one of the basic quantum gates used in quantum computing, with the matrix form:

$$Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$$

Action on Qubit States:

- $Y|0\rangle = i|1\rangle$
- $Y|1\rangle = -i|0\rangle$

This gate behaves similarly to the Pauli-X (bit-flip) gate, but it introduces a complex phase rotation.

4.7.4 The ZZ Interaction Gate and Basis Rotation

The **ZZ gate** is a two-qubit gate that induces a phase shift depending on the $Z \otimes Z$ interaction between qubits. It's essential in simulating spin-spin interactions, such as those in the TFIM.

- Used to implement the Ising coupling between neighboring spins.
- Often combined with single-qubit X or H gates to model transverse fields.

Basis Rotation: This refers to transforming qubits into a computational basis that makes certain interactions — like nearest-neighbor couplings — easier to implement.

”Custom unitary gates are gates not included in the standard Qiskit gate set and are defined by the user using unitary matrices to achieve specialized quantum transformations.”

- **Exact Diagonalization, Density Matrix Renormalization Group (DMRG), and Monte Carlo methods** are widely used numerical techniques for simulating spin-based Hamiltonians in quantum systems. These approaches provide insight into ground state properties, phase transitions, and quantum correlations in many-body systems.[\[11\]](#)

Chapter 5

Hamiltonian Simulation of CoNb_2O_6

5.1 Define the Physical System (1D TFIM)

CoNb_2O_6 (Cobalt Niobate) is a quasi-one-dimensional (1D) quantum magnetic material recognized for exhibiting quantum phase transitions. Its structure consists of linear chains of Co^{2+} ions, each carrying an effective spin- $\frac{1}{2}$. These spins interact dominantly along the chain direction via nearest-neighbor coupling, while inter-chain interactions are weak, justifying the 1D approximation.

The system exhibits strong uniaxial anisotropy due to crystal field effects and spin-orbit coupling, enforcing spins to align predominantly along the z -axis. This intrinsic anisotropy satisfies the key condition of the Ising model, where spins prefer alignment along a single axis. The Ising-like behavior arises from the dominant interaction along the z -axis ($Z_i Z_{i+1}$), which energetically favors spin alignment or anti-alignment.

The Transverse Field Ising Model (TFIM) is a variation of the standard Ising model where a magnetic field is applied perpendicular to the direction of spin quantization.

A transverse magnetic field applied along the x -direction introduces quantum fluc-

tuations. This field competes with the Ising interaction, enabling spins to flip between up and down states, driving the system toward a quantum phase transition from an ordered ferromagnetic phase to a disordered quantum paramagnetic phase as the transverse field increases.^[5]

5.2 Wave Function Representation

For an N -spin chain, the Hilbert space spans 2^N dimensions. Each basis state represents a distinct spin configuration in the computational basis $|0\rangle$ (spin up) and $|1\rangle$ (spin down) along the z -axis. The wave function is expressed as:

$$|\Psi\rangle = \sum_{i=1}^{2^N} c_i |i\rangle$$

where $|i\rangle$ are computational basis states and c_i are complex amplitudes. Typical initial states include the ground state of the transverse field term or a fully aligned state like $|000 \dots 0\rangle$.^[17]

5.3 Quantum Ising Hamiltonian for CoNb_2O_6

The Hamiltonian representing CoNb_2O_6 in the 1D TFIM framework is written using Pauli operators:

$$H = -J \sum_i Z_i Z_{i+1} - h \sum_i X_i$$

where:

- J is the interaction strength along the z -direction (Ising coupling).
- h is the transverse magnetic field strength along the x -axis.
- Z_i and X_i are Pauli-Z and Pauli-X operators acting on qubit i .
- **Quantum Phase Transition:** In the transverse field Ising model, the nature of the system's ground state depends on the relative strength of the interaction terms in the Hamiltonian. When the transverse field strength h is small, the system resides in an ordered phase, where the spins tend to align along the z -axis due to the dominant Ising interaction term. As h increases, quantum fluctuations become significant, driven by the transverse field, and the system undergoes a transition into a disordered phase where spin alignment is lost.

Interaction Terms:

- **Ising Coupling** ($Z_i Z_{i+1}$): Enforces spin alignment or anti-alignment along the z -axis. This term embodies the classical Ising behavior intrinsic to CoNb_2O_6 .
- **Transverse Field** (X_i): Facilitates quantum spin flips, driving transitions between $|0\rangle$ and $|1\rangle$ states.

This competition between classical ordering (from $Z_i Z_{i+1}$ coupling) and quantum fluctuations (from the transverse field) is fundamental to the TFIM description of CoNb_2O_6 .

5.3.1 Limiting Cases of the Transverse Field Ising Model

The behavior of the transverse field Ising model (TFIM) is strongly influenced by the relative strengths of the interaction parameter J and the transverse magnetic field h . Two key regimes are:

1. **Strong Interaction Regime** ($J \gg h$): The Hamiltonian is given by:

$$H = -J \sum_{\langle i,j \rangle} S_i^z S_j^z - h \sum_i S_i^x$$

In this regime, the Ising interaction term $S_i^z S_j^z$ dominates. Spins energetically prefer to align along the z -axis, resulting in a classically ordered phase where all spins are either up ($\uparrow\uparrow$) or down ($\downarrow\downarrow$). The transverse field term S_i^x , which tends to flip spins, is too weak to disrupt this order. Hence, the ground state exhibits z -axis spin alignment.

2. **Strong Field Regime** ($h \gg J$): In this case, the transverse field term dominates. The interaction term becomes negligible, and spins tend to align along the x -axis. This leads to a disordered quantum phase known as a *quantum paramagnet*, where each spin exists in a superposition state pointing in the x -direction. The Ising interaction cannot maintain z -axis order, and thus the system no longer exhibits classical spin alignment.

5.3.2 Fundamental Criteria for realizing the Ising Model

To successfully describe a material or spin system using the Ising model, certain key physical characteristics must be met. These ensure that the assumptions of the model are valid in the given physical context.[\[13\]](#)

Table 5.1: Essential Physical Properties for Ising Model Systems

Property	Description	Why It's Necessary
Two-level spin system	Each spin has two accessible states (e.g., spin- $\frac{1}{2}$ or up/down ion)	Enables binary modeling using $S_z = \pm 1$ or $\pm \frac{1}{2}$
Uniaxial anisotropy	System favors alignment along a fixed axis (usually z)	Suppresses transverse spin terms; maintains Ising constraint
Short-range exchange	Spins interact predominantly with nearest neighbors	Matches model assumption of local interactions
Spin alignment energy cost	Distinct energy for aligned ($-J$) and anti-aligned ($+J$) spins	Drives phase transitions and ordering behavior
Quantum fluctuations (optional)	Enabled via transverse magnetic field	Allows tunneling between spin states in quantum Ising variants

5.4 Hamiltonian Decomposition

The Hamiltonian is decomposed into two non-commuting components for circuit implementation:

Interaction Term (H_Z):

$$H_Z = -J \sum_i Z_i Z_{i+1}$$

This term models the dominant spin-spin interaction along the z -axis responsible for the Ising behavior.

Transverse Field Term (H_X):

$$H_X = -h \sum_i X_i$$

This term introduces quantum fluctuations through spin flips along the x -axis, counteracting the ordering from H_Z .[\[15\]](#)

5.5 Mapping to Pauli Operators

The Hamiltonian is explicitly expressed in the Pauli gate formalism, making it directly translatable into quantum circuits:

- $Z_i Z_{i+1} \rightarrow$ Two-qubit $Z \otimes Z$ interactions.
- $X_i \rightarrow$ Single-qubit X rotations.

These terms are natural in the gate-based model of quantum computing and align with Qiskit's native operations.[\[26\]](#)

5.6 Applying Suzuki-Trotter Decomposition

The time evolution operator is:

$$U(t) = e^{-iHt}$$

Given that H_Z and H_X do not commute, Suzuki-Trotter decomposition approximates this as:

$$U(t) \approx \left(e^{-iH_Z \Delta t} e^{-iH_X \Delta t} \right)^n$$

where n is the number of Trotter steps and $\Delta t = t/n$.

- H_Z : Implemented via two-qubit ZZ rotations.
- H_X : Implemented via single-qubit X -axis rotations.

There is also one more method if we don't want to undergo Suzuki-Trotter Decomposition;

QDrift is a method to simulate Hamiltonians, especially for quantum systems like molecules or spin chains.[\[2\]](#)

5.7 Mapping to Basic Quantum Gates

- **ZZ interactions:**

$$Z_i Z_{i+1} \Rightarrow RZZ(\theta), \quad \theta = 2J\Delta t$$

- **X field:**

$$X_i \Rightarrow Rx(2h\Delta t)$$

These mappings efficiently translate the physical Hamiltonian into quantum gates suitable for current NISQ hardware.

5.8 Quantum Circuit Construction

The full quantum circuit alternates between layers of:

1. **RZZ gates** for nearest-neighbor ZZ interactions.
2. **Rx rotations** on each qubit for the transverse field effect.
3. Repeat this sequence for n Trotter steps.

This structure simulates the dynamics of the CoNb_2O_6 Hamiltonian under the 1D TFIM framework.

5.9 Simulation on Real IBM Quantum Computers Using Qiskit

Simulation involves deploying these circuits on IBM Quantum hardware. Key procedures include:

- Utilizing Qiskit primitives such as `SamplerV2` and `EstimatorV2`.
- Applying noise mitigation strategies.
- Comparing quantum hardware outcomes with classical simulations.
- Measuring observables like ground state energy, magnetization, and correlation functions.

5.10 Comparison of Classical and Quantum Simulation Results

Classical Methods:

- Use of exact diagonalization and matrix exponentiation.

Quantum Methods:

- Implementation of Trotterized circuits subject to hardware noise and sampling errors.

Key Observations:

- Quantum simulations scale more favorably in memory but face limitations due to decoherence.
- Classical simulations become infeasible for larger systems but are deterministic and noise-free.[\[12\]](#)

Conclusion:

The quantum simulation of CoNb_2O_6 under the TFIM model captures the balance between classical ordering (Z-Z interactions) and quantum fluctuations (X-field). This framework demonstrates the ability of quantum computers to simulate real condensed matter systems, offering promising results within current hardware limitations.

References

- [1] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, “Quantum machine learning,” *Nature*, vol. 549, no. 7671, pp. 195–202, 2017.
- [2] E. Campbell, “Random compiler for fast hamiltonian simulation,” *Physical Review Letters*, vol. 123, no. 7, p. 070 503, 2019. DOI: [10.1103/PhysRevLett.123.070503](https://doi.org/10.1103/PhysRevLett.123.070503).
- [3] J. Carrasquilla and R. G. Melko, “Machine learning phases of matter,” *Nature Physics*, vol. 13, no. 5, pp. 431–434, 2017.
- [4] C. Chamberland, G. H. L. Zhu, T. J. Yoder, K. M. Svore, and A. W. Cross, “Building a fault-tolerant quantum computer using concatenated cat codes,” *Nature*, vol. 595, pp. 383–387, 2022. DOI: [10.1038/s41586-021-03588-y](https://doi.org/10.1038/s41586-021-03588-y).
- [5] R. Coldea, D. Tennant, E. Wheeler, *et al.*, “Quantum criticality in an ising chain: Experimental evidence for emergent e8 symmetry,” *Science*, vol. 327, no. 5962, pp. 177–180, 2010. DOI: [10.1126/science.1180085](https://doi.org/10.1126/science.1180085).
- [6] A. Dutta, G. Aeppli, B. K. Chakrabarti, U. Divakaran, T. F. Rosenbaum, and D. Sen, *Quantum Phase Transitions in Transverse Field Spin Models: From Statistical Physics to Quantum Information*. Cambridge University Press, 2015.
- [7] E. Farhi, J. Goldstone, and S. Gutmann, “A quantum approximate optimization algorithm,” *arXiv preprint arXiv:1411.4028*, 2014.
- [8] R. P. Feynman, “Simulating physics with computers,” *International journal of theoretical physics*, vol. 21, no. 6-7, pp. 467–488, 1982. DOI: [10.1007/BF02650179](https://doi.org/10.1007/BF02650179).
- [9] I. M. Georgescu, S. Ashhab, and F. Nori, “Quantum simulation,” *Reviews of Modern Physics*, vol. 86, no. 1, pp. 153–185, 2014.

-
- [10] N. Gisin, G. Ribordy, W. Tittel, and H. Zbinden, “Quantum cryptography,” *Reviews of Modern Physics*, vol. 74, no. 1, pp. 145–195, 2002.
- [11] C. Hempel, C. Maier, J. Romero, *et al.*, “Quantum chemistry calculations on a trapped-ion quantum simulator,” *Physical Review X*, vol. 8, no. 3, p. 031 022, 2018. DOI: [10.1103/PhysRevX.8.031022](https://doi.org/10.1103/PhysRevX.8.031022).
- [12] Y. e. a. Kim, “Evidence for the utility of quantum computing before fault tolerance,” *Nature*, vol. 618, no. 7965, pp. 500–505, 2023.
- [13] A. W. Kinross, W. T. Fuhrman, C. S. Nelson, *et al.*, “Evolution of quantum fluctuations near the quantum critical point of the transverse field ising chain system conb_2o_6 ,” *Physical Review X*, vol. 4, no. 3, p. 031 008, 2014. DOI: [10.1103/PhysRevX.4.031008](https://doi.org/10.1103/PhysRevX.4.031008).
- [14] T. D. Ladd, F. Jelezko, R. Laflamme, Y. Nakamura, C. Monroe, and J. L. O’Brien, “Quantum computers,” *Nature*, vol. 464, no. 7285, pp. 45–53, 2010. DOI: [10.1038/nature08812](https://doi.org/10.1038/nature08812).
- [15] S. Lloyd, “Universal quantum simulators,” *Science*, vol. 273, no. 5278, pp. 1073–1078, 1996.
- [16] A. Montanaro, “Quantum algorithms: An overview,” *npj Quantum Information*, vol. 2, no. 1, p. 15 023, 2016. DOI: [10.1038/npjqi.2015.23](https://doi.org/10.1038/npjqi.2015.23).
- [17] Nelson and Chung, *Quantum Computing and Information*. Unknown Publisher, 2020.
- [18] J. Preskill, “Quantum computing in the nisc era and beyond,” *Quantum*, vol. 2, p. 79, 2018. DOI: [10.22331/q-2018-08-06-79](https://doi.org/10.22331/q-2018-08-06-79). [Online]. Available: <https://quantum-journal.org/papers/q-2018-08-06-79/>.
- [19] Qiskit Development Team, *Qiskit primer: Learn quantum computation using qiskit*, Accessed: 2025-07-05, IBM Quantum, 2023. [Online]. Available: <https://qiskit.org/learn/primer>.
- [20] Qiskit Development Team, *Qiskit: An open-source framework for quantum computing*, Accessed: 2025-07-05, IBM Quantum, 2023. [Online]. Available: <https://qiskit.org/documentation/>.

- [21] Qiskit Development Team, *Qiskit installation guide and system requirements*, Accessed: 2025-07-05, IBM Quantum, 2024. [Online]. Available: https://qiskit.org/documentation/getting_started.html.
- [22] J. D. H. Ryan, *Quantum Computing: An Applied Approach*, 2nd. Springer, 2021, ISBN: 9783030239213.
- [23] S. Sachdev, *Quantum Phase Transitions*. Cambridge University Press, 2011.
- [24] Smite-Meister, *Bloch sphere: A geometrical representation of a two-level quantum system*, https://commons.wikimedia.org/wiki/File:Bloch_sphere.svg, Own work. Licensed under CC BY-SA 3.0., n.d.
- [25] The Quantum Insider. “Quantum computing roadmaps: A look at the maps and predictions of major quantum players.” Accessed: 2025-06-30. (May 2025), [Online]. Available: <https://thequantuminsider.com/2025/05/16/quantum-computing-roadmaps-a-look-at-the-maps-and-predictions-of-major-quantum-players/>.
- [26] J. D. Whitfield, J. Biamonte, and A. Aspuru-Guzik, “Simulation of electronic structure hamiltonians using quantum computers,” *Molecular Physics*, vol. 109, no. 5, pp. 735–750, 2011. DOI: [10.1080/00268976.2011.552441](https://doi.org/10.1080/00268976.2011.552441).
- [27] N. S. Yanofsky and M. A. Mannucci, *Quantum Computing for Computer Scientists*. Cambridge University Press, 2008, ISBN: 9780521879965.

Appendix A

Appendix: Qiskit Implementation and Python Code

A.1 Bell State

```
from qiskit import QuantumCircuit

qc = QuantumCircuit(2)

qc.h(0)

qc.cx(0, 1)

qc.draw(output='mpl')
```