UNIVERSITÀ DEGLI STUDI DI GENOVA

DIPARTIMENTO DI FISICA

UNIVERSITÀ
DEGLI STUDI
DI GENOVA

Degree Course in Physics

Quantum Non-Demolition Measurement Techniques for Gradient Evaluation and Resource Reduction in Variational Quantum Algorithms

Supervisor:
Hon. Prof. Paolo Solinas

Candidate:
Rabia Abdul Razaq

ANNO ACCADEMICO 2023/2026

To my ever-supportive and respected parents.

ACKNOWLEDGEMENTS

I would like to express my gratitude to everyone who helped me throughout the journey of this research. This work represents not only an academic accomplishment but also a journey supported by the guidance, advice, and encouragement provided by many people. First and foremost, I would like to thank my kind and supportive supervisor, Professor Paolo Solinas, for all of his encouragement, guidance, and understanding during this research. His deep knowledge, constant support, and insightful feedback have greatly improved the quality of this work. I am deeply grateful for the opportunity to learn under his supervision. I would also like to extend my heartfelt gratitude to my coordinator, Mr. Silvano Tosi, whose generosity, unwavering support, and thoughtful guidance through every challenge have been truly invaluable. His assistance has played an essential role in making my time here meaningful and successful. I am very grateful to my husband, Muhammad Rashid, who has always been there for me, with patience and understanding, along this journey. My biggest source of strength has been his belief in me and his constant support. I also want to express my sincere gratitude to my parents for their unwavering love, unending support, and many sacrifices, all of which allowed me to finish this journey. They have always motivated me by having faith in my abilities. My gratitude also extends to my colleagues, Giovanni Minuto and Dario Malegari, for their continuous support and stimulating discussions. Their ideas, perspectives, and professional guidance were extremely valuable in improving this research and expanding my knowledge of the topic. Finally, I would like to express my gratitude to all those whose names I may not have mentioned but who supported me in any way throughout this journey.

ABSTRACT

Quantum computing has become a potential way to solve complex problems that conventional computers can't handle, especially those involving strongly correlated quantum systems. Classical simulation of these systems becomes exponentially more challenging as the number of interacting particles rises, resulting in significant computational and memory limitations. As a result, classical computational approaches are unable to handle many significant physical and chemical systems, including molecular structures, biological molecules, and quantum materials. Quantum computers operate using the same quantum mechanical principles as these systems. Consequently, they provide a natural framework for exploring large Hilbert spaces and capturing complex quantum correlations.

This thesis begins by discussing the fundamental principles of quantum mechanics that are applicable to computation, such as qubits, superposition, and entanglement. The construction of quantum circuits is discussed after a discussion of basic quantum gates and their matrix representations. These ideas are the foundation of quantum algorithms and offer the essential framework for comprehending the modeling and manipulation of complicated quantum states using quantum hardware.

Building on this foundation, the thesis focuses on Variational Quantum Algorithms (VQAs), which offer a feasible hybrid quantum-classical approach to optimizing near-term quantum devices. In VQAs, a parameterized quantum circuit is used to approximate a ground state energy of the quantum system. A classical optimizer then iteratively updates the variational parameters to minimize a cost function, usually the expectation value of the system's Hamiltonian. A thorough analysis of the function of gradient-based optimization is conducted, encompassing the parameter-shift rule that is employed to calculate precise energy derivatives. The discussion is expanded to include major applications such as the Quantum Approximate Optimization Algorithm and developing approaches to quantum machine learning, emphasizing the importance of efficient and

accurate gradient evaluation.

A major challenge in the real-world application of VQAs is the high resource cost of gradient estimation using conventional direct measurement techniques. These techniques require repeated projective measurements and several executions of parameter-shifted quantum circuits, which lead to significant measurement overhead and deeper circuits. Quantum Non-Demolition Measurement (QNDM) is examined in this thesis as a potential framework for gradient evaluation in order to get around this limitation. Based on recent theoretical advances, QNDM introduces an ancillary detector qubit that coherently stores gradient information within its phase. Derivatives can then be estimated indirectly through detector measurements. This allows gradient information to be extracted without destroying the quantum state and with a greatly reduced number of measurements.

The efficiency of QNDM is evaluated by extending the study to systems with bigger Hamiltonians and greater molecular complexity, as well as by comparing it in detail with direct measurement techniques utilizing benchmark molecular systems like LiH, Li_2, and H_2. Although both methods successfully converge to the ground-state energy, as shown by numerical results, QNDM consistently uses fewer quantum resources in terms of logical gate operations, measurement shots, and circuit executions. Additionally, as system size grows, QNDM's resource advantage becomes more noticeable, suggesting that it has favorable scalability for bigger molecular systems.

This thesis concludes that, while both direct measurement and quantum non-demolition measurement approaches allow for precise variational optimization, QNDM gives a significant decrease in quantum resources. These results establish QNDM as an effective and scalable method for developing quantum simulations of realistic physical and chemical systems and for executing variational quantum algorithms on near-term quantum hardware.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

FOUNDATIONS OF QUANTUM COMPUTATION

This chapter provides an overview of the basic ideas of quantum computing by describing the differences between quantum and classical computing before suddenly introducing all of the mathematical tools that comprise these fundamental components of quantum computing.

1.1 Exploring the Quantum Computing World

From human calculation until the age of supercomputers, computation has undergone an interesting progression. The physical limitations of conventional computers have been revealed in recent years, though, and this has prompted studies into new areas, such as quantum computing. The focus of this introductory section is the reasons for developing quantum computing devices. Even though classical computers have advanced significantly, some problems still require prohibitively long computation times, like factorizing big numbers for cryptographic applications (like the RSA protocol) [1] or globally optimizing complex energy landscapes. This dilemma results from the sequential structure of classical computation, which might take an exponentially long time to solve certain complex problems. A series of extensible algebraic computations is based on the fundamental process of multiplying two extremely big prime numbers, which is the foundation of the venerable RSA protocol [1]. It is currently well known that, barring implementation

errors, there is very little chance that a traditional computer could effectively crack the RSA protocol. As explained by Shor in 1994, one of the most significant challenges to the RSA protocol is Shor's algorithm, which indicates that quantum computers are capable of factoring prime integers effectively.

Quantum mechanics theory was developed in the early 1900s to describe how atoms and subatomic particles act. The principles of quantum mechanics are used in quantum computing to transcend the inherent constraints of conventional computing. To put it another way, many interesting issues are practically hard to answer on a traditional computer, not because they are impossible to solve, but rather because the computation requires a large number of traditional resources. Concepts like **superposition** and **entanglement** [2], which are central to quantum computing and give it more computational power than classical computation for some problems. These concepts are also foundational principles of quantum mechanics and challenge our intuitive understanding of the classical physical world. The presence of non-local quantum correlations, which seem to defy the no-signaling principle of special relativity (that information cannot be communicated more quickly than light), is one of the strange outcomes that arise from these two characteristics and appear to go against our physical intuitions.

These factors gave rise to the well-known Einstein–Podolsky–Rosen paradox, which raised serious doubts about the accuracy of quantum mechanics and signaled a turning point in the discussion of the nature of physical reality. From maximizing industrial processes to high-energy physics [3], simulating molecular systems for drug discovery (quantum chemistry), finance [4], and achieving the so-called quantum advantage in particular research fields, quantum computing might offer solutions for problems that traditional methods are unable to address. Although we are only beginning to see practical implementations, quantum computing has special obstacles, including large-scale implementation and handling of quantum faults (quantum error correction). In quantum computing, the qubit is the most basic unit of information, while it is the bit in the case of classical computing. From a physical perspective, the bit, a system with two states, is a representation of the logical values 0 and 1. However, the situation differs fundamentally in quantum mechanics. A qubit can exist not just in the two states $|0\rangle$ and $|1\rangle$, but also in a linear superposition of these states; in other words, the qubits most general state is

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{1.1}$$

There are several ways to physically realize a qubit, including employing photons, ions (ion-trapped devices), or superconducting circuits (superconducting devices). However, we first describe the mathematical foundation of quantum computation before moving on to the technologies.

1.2 Quantum Mechanics Basics for Quantum Computing

We suggest offering a more exact description of quantum mechanics with finite-dimensional state spaces in this section. A collection of postulates that explain how quantum systems behave forms the basis of quantum mechanics.

1.2.1 First Postulate: State Representation in Hilbert Space

The state of a quantum system can be described by a vector in a Hilbert space equipped with an inner product. The Hilbert space is a complex two-dimensional vector space, and the qubit state for a single qubit is a linear combination of the basis states ($|0\rangle$) and ($|1\rangle$). Each physical system has a corresponding Hilbert space H, where valid states are normalized vectors satisfying $|\langle\psi|\psi\rangle|^2 = 1$.

1.2.2 Second Postulate: Measurement in Quantum Mechanics

In quantum mechanics, measurements are described by Hermitian operators. A Hermitian operator $\hat{A}$ is linked to any measurable physical quantity A, and its eigenvalues a_n satisfy

$$\hat{A}|a_n\rangle = a_n|a_n\rangle, \tag{1.2}$$

with corresponding eigenvector $|a_n\rangle$. If the system's state

$$|\psi\rangle = \sum_n c_n|a_n\rangle$$

then the probability $P(a_n)$ of achieving the measurement result a_n is

$$P(a_n) = |C_n|^2 \tag{1.3}$$

where a_n forms a complete orthonormal basis of $\mathcal{H}$. For the state $|\psi\rangle$, the expected value of the observable $\hat{A}$ on the state $|\psi\rangle$ is given by $\langle A\rangle = \langle\psi|A|\psi\rangle = \mathrm{Tr}(A|\psi\rangle\langle\psi|)$.

According to the computational basis, a single qubit in the state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ yields the outcomes $|0\rangle$ with probability $|\alpha|^2$ and $|1\rangle$ with probability $|\beta|^2$.

1.2.3 Third Postulate: Quantum System Time Evolution in a Closed System

A unitary time evolution operator $U(t, t_0)$ describes how a quantum system changes over time from an initial time t_0 to a time t. A unitary operator U has the property $U^\dagger(t, t_0)U(t, t_0) = U(t, t_0)U^\dagger(t, t_0) = I$, where $U^\dagger$ is the conjugate transpose of U, and I is the identity operator, according to Nielsen and Chung's book [5]. Since the initial state of a quantum system at time t_0 is $|\psi(t_0)\rangle = |\psi_0\rangle$, then at time t, we will have $|\psi(t)\rangle = |\psi_t\rangle$ given by $|\psi_t\rangle = U(t, t_0)|\psi_0\rangle$. In quantum computing, unitary operators are quantum gates that modify the states of qubits.

1.3 The qubit:

The core idea behind classical information and computation is the bit, the smallest unit of data that can represent either a 0 or a 1. Conversely, a qubit can at the same time be a superposition of 0 and 1. While classical bit measurements do not change their state, measuring a qubit collapses its superposition. One bit of information can be encoded by a single qubit, while superdense coding allows it to store up to two bits. While a set of n bits can only be in one of the 2^n conceivable states, one of two states is always present in a classical bit. A 2D complex vector or coordinate in quantum space describes the state of a qubit in a superposition, which has a non-zero probability of being both states at the same time and takes two complex numbers to describe.

1.3.1 Standard representation:

A qubits general quantum state in quantum mechanics is defined as a linear combination of two orthonormal basis states, also known as basis vectors. Typically, these vectors are represented as

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{1.4}$$

these two orthonormal basis states, $|0\rangle$ and $|1\rangle$, are referred to as the computational basis states. It is possible to mix qubit basis states to create product basis states. In 2^n dimensional Hilbert space, a superposition state vector typically represents n qubits.

1.3.2 Qubit states

A pure qubit state is characterized by the coherent superposition of basis states. A single qubit ψ can be expressed as a linear combination of $|0\rangle$ and $|1\rangle$

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{1.5}$$

where α and β complex numbers. According to the Born rule, if we measure this qubit using the standard basis, the probability of $|0\rangle$ with value 0 is $|\alpha|^2$, and the probability of $|1\rangle$ with value 1 is $|\beta|^2$. The second postulate of probability theory must be satisfied by the following equation because the absolute squares of amplitudes produce probabilities α and β

$$|\alpha|^2 + |\beta|^2 = 1. \tag{1.6}$$

In other words, a vector in complex Hilbert space with two dimensions represents a qubits state.

1.3.3 Bloch sphere representation

At first sight, it may appear that $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ has four degrees of freedom because both α and β are complex numbers with two degrees of freedom. The normalizing constraint $|\alpha|^2 + |\beta|^2 = 1$ removes one degree of freedom. This indicates that by changing the coordinates appropriately, one of the degrees of freedom can be eliminated. One possible representation uses Hopf coordinates.

$$\begin{aligned} \alpha &= e^{i\delta} \cos\frac{\theta}{2}, \\ \beta &= e^{i(\delta+\varphi)} \sin\frac{\theta}{2}. \end{aligned} \tag{1.7}$$

In addition, we can just pick α to be real (or β in the case where α is zero) for the one-qubit system with only two remaining degrees of freedom since there is no physical meaning to the global phase of the state $e^{i\delta}$.

$$\alpha = \cos\frac{\theta}{2}, \tag{1.8}$$

$$\beta = e^{i\varphi} \sin\frac{\theta}{2}. \tag{1.9}$$

A single qubits possible states can be seen using a Bloch sphere [6]. On the sphere's surface, a qubit can be found anywhere, whereas a traditional bit can only exist at the "North Pole" or "South Pole," represented by the symbols $|0\rangle$ and $|1\rangle$. An example of a

state on the equator is $(|0\rangle + |1\rangle)/\sqrt{2}$. The surface of the Bloch sphere is a representation of each pure qubit state, which is defined by two angles: θ and φ. Only at the poles does a qubit act like a classical bit in the classical limit. The Bloch sphere can be shown in the way as described in Ref.[6]:

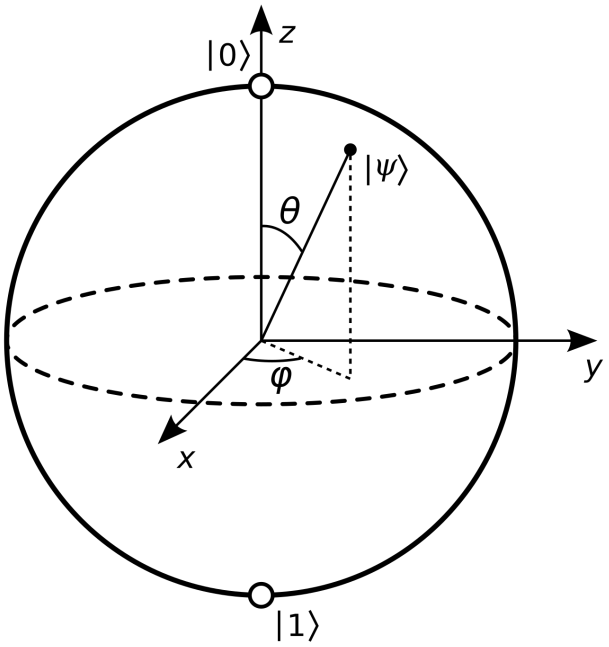


Figure 1.1: Bloch sphere: The poles represent the states $|0\rangle$ and $|1\rangle$, whereas the remaining points represent superpositions of the basis states. The Bloch sphere representation defines orthogonal states as antipodal points on the sphere. Figure from Ref. [6].

Multiple Qubits:

A system with n bits in classical computing can only represent one potential state at a time. However, n qubits exist in a superposition of classical states in quantum computing. A multi-qubit system is made up of the tensor product of each qubit state. Two basis states are superposed to form a single qubit. A system of n qubits spans a 2^n Hilbert space, with each basis state corresponding to a classical bitstring. An n qubit system possesses a computational basis consisting of two basis states, which we shall label

$$|i_1, i_2 \ldots, i_n\rangle = |i_1\rangle \otimes |i_2\rangle \otimes \cdots \otimes |i_n\rangle, \quad \text{where } i_k \in \{0, 1\}. \tag{1.10}$$

A general n qubit quantum state is represented as

$$|\psi\rangle = \sum_{i_1, i_2, \ldots, i_n} c_{i_1, i_2 \ldots, i_n} |i_1, i_2 \ldots, i_n\rangle. \tag{1.11}$$

where the complex probability amplitudes $c_{i_1,i_2\ldots,i_n}$ satisfy the normalization condition:

$$\sum_{i_1,i_2,\ldots,i_n} |c_{i_1,i_2\ldots,i_n}|^2 = 1. \tag{1.12}$$

For instance, the computational basis states of a two-qubit system are represented in a four-dimensional Hilbert space as follows:

$$|00\rangle, |01\rangle, |10\rangle, |11\rangle \tag{1.13}$$

A linear combination of basic states can be used to represent a general 2-qubit system:

$$|\psi\rangle = c_{00}|00\rangle + c_{01}|01\rangle + c_{10}|10\rangle + c_{11}|11\rangle. \tag{1.14}$$

under the normalizing requirement, the total squared magnitudes of the coefficients must equal one.

$$|c_{00}|^2 + |c_{01}|^2 + |c_{10}|^2 + |c_{11}|^2 = 1. \tag{1.15}$$

A four-dimensional column vector can be used to represent each basis state in the computational basis:

$$\begin{aligned} |00\rangle &= \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad |01\rangle = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \\ |10\rangle &= \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \quad |11\rangle = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}. \end{aligned} \tag{1.16}$$

As the number of qubits rises, the state space correspondingly grows, allowing for quantum parallelism and making multi-qubit systems essential in quantum processing.

1.4 Quantum Entanglement

Entanglement is a basic idea that distinguishes quantum mechanics from classical physics. Entanglement, along with the concept of superposition, enables quantum systems to exhibit behaviors that classical models cannot replicate. These properties serve as the foundation for quantum computation and information processing, enabling increased

parallelism and new algorithms that outperform their classical counterparts in terms of capacity and efficiency.

To demonstrate this, let us imagine two qubits, q_A and q_B, that are part of the Hilbert spaces H_A and H_B, respectively. The combined system is described by the tensor product of the Hilbert spaces, $\mathcal{H}_\mathcal{A}\otimes\mathcal{H}_\mathcal{B}$. The individual qubit basis $\{\,|0\rangle_A, |1\rangle_A\,\}$ and $\{\,|0\rangle_B, |1\rangle_B\,\}$ is used to derive the combined basis, specifically provided by $\{\,|0_A\rangle\otimes|0_B\rangle,\ |0_A\rangle\otimes|1_B\rangle,\ |1_A\rangle\otimes|0_B\rangle,\ |1_A\rangle\otimes|1_B\rangle\,\}$, i.e.:

$$|\mathcal{X}_{ij}\rangle = |i_A\rangle \otimes |j_B\rangle \tag{1.17}$$

The most general states of the two qubits in their respective Hilbert spaces are:

$$|\psi_A\rangle = \alpha_A|0_A\rangle + \beta_A|1_A\rangle \tag{1.18}$$

$$|\psi_B\rangle = \alpha_B|0_B\rangle + \beta_B|1_B\rangle \tag{1.19}$$

Then, the tensor product gives the composite system's basic state as:

$$|\psi\rangle = |\psi_A\rangle \otimes |\psi_B\rangle \tag{1.20}$$

$$= \alpha_A\alpha_B|0_A\rangle \otimes |0_B\rangle + \alpha_A\beta_B|0_A\rangle \otimes |1_B\rangle + \beta_A\alpha_B|1_A\rangle \otimes |0_B\rangle + \beta_A\beta_B|1_A\rangle \otimes |1_B\rangle \tag{1.21}$$

$$= \alpha_{00}|\chi_{00}\rangle + \alpha_{01}|\chi_{01}\rangle + \alpha_{10}|\chi_{10}\rangle + \alpha_{11}|\chi_{11}\rangle \tag{1.22}$$

A composite quantum state can be categorized as follows:

Separable if it can be defined as a tensor product of two different states of specific subsystems:

$$|\psi\rangle = |\psi_A\rangle \otimes |\psi_B\rangle.$$

Here, the coefficient satisfy the relationship

$$\alpha_{00}\alpha_{11} = \alpha_{01}\alpha_{10}.$$

Entangled if it is not possible to express it in such a factorized way. This happens when the coefficients do not satisfy the separability condition, i.e.,

$$\alpha_{00}\alpha_{11} \neq \alpha_{01}\alpha_{10}.$$

Entangled states are non-classical correlations between the subsystems. Unlike separable states, they can not be decomposed into individual descriptions of the constituent qubits. These correlations are the resources behind quantum communication protocols like teleportation, superdense coding, and quantum cryptography and most of the quantum algorithms.

1.5 QUANTUM GATES AND CIRCUITS:

Quantum computing is based on a complicated interaction between quantum gates and circuits, not much unlike the logic gates and circuits in classical computers. This chapter explores the intricate realm of quantum gates, their role in manipulating qubits, and the construction of quantum circuits that orchestrate these gates to perform complicated computations.

Single-Qubit Gates

A single qubits state is altered by single-qubit gates while maintaining its superposition or entanglement characteristics. To manipulate quantum states, a set of one-qubit operations is known as Pauli gates. These Pauli gates are represented as Pauli-X, Pauli-Y, and Pauli-Z gates, which are important parts of more complex quantum algorithms and circuits. They enabled manipulation of states and offer a platform for performing more complex quantum operations. According to the computational basis (1.4), the Pauli matrices σ_x, σ_y, and σ_z are based on the 2×2 unitary matrices Ref.[5].

Pauli-X Gate: This gate flipped the state of the qubit from $|0\rangle$ to $|1\rangle$ and $|1\rangle$ to $|0\rangle$, just like the classical NOT gate does. Its 2×2 matrix in the computational basis (1.4) can be written like this:

$$X|0\rangle = |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad X|1\rangle = |0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}.$$

Pauli-Y Gate: The qubit undergoes both a bit and a phase flip by the Y gate; it is a hybrid of the X and Z gates. In the computational basis $\{|0\rangle, |1\rangle\}$ for one qubit, the Pauli-Y gate is

$$Y|0\rangle = i|1\rangle = \begin{bmatrix} 0 \\ i \end{bmatrix}, \quad Y|1\rangle = -i|0\rangle = \begin{bmatrix} -i \\ 0 \end{bmatrix}, \quad Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}.$$

Pauli-Z Gate: The Z gate flipped the qubit state's phase, while the basic states $|0\rangle$ and $|1\rangle$ remain unaltered, but inverting the relative phase of $|1\rangle$. In the computational basis, the Pauli-Z gate for a single qubit is

$$Z|0\rangle = |0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad Z|1\rangle = -|1\rangle = \begin{bmatrix} 0 \\ -1 \end{bmatrix}, \quad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}.$$

Another basic class of quantum gates is produced by exponentiating the Pauli matrices, this leads to parameterized rotations along the x, y, and z axes of the Bloch sphere, also known as Pauli rotation gates:

- **Pauli-X as a Parameterized Gate:** A rotation about the X-axis is represented by the X gate as $\boldsymbol{\theta} = \pi$

$$R_x(\pi) = e^{-i\frac{\pi}{2}X}$$

The generic X-Rotation gate can be represented as:

$$R_x(\theta) = e^{-i\frac{\theta}{2}X} = \begin{bmatrix} \cos(\theta/2) & -i\sin(\theta/2) \\ -i\sin(\theta/2) & \cos(\theta/2) \end{bmatrix} \tag{1.23}$$

- **Pauli-Y as a Parameterized Gate:** The rotation of the Y-axis by the Y gate is equivalent to $\boldsymbol{\theta} = \pi$

$$R_y(\pi) = e^{-i\frac{\pi}{2}Y}$$

The general Y-Rotation gate can be expressed as

$$R_y(\theta) = e^{-i\frac{\theta}{2}Y} = \begin{bmatrix} \cos(\theta/2) & -\sin(\theta/2) \\ \sin(\theta/2) & \cos(\theta/2) \end{bmatrix} \tag{1.24}$$

- **Pauli-Z as a Parameterized Gate:** An Z-axis rotation is represented by the Z gate as $\boldsymbol{\theta} = \pi$

$$R_z(\pi) = e^{-i\frac{\pi}{2}Z}$$

Z-Rotation gate can be expressed as:

$$R_z(\theta) = e^{-i\frac{\theta}{2}Z} = \begin{bmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{bmatrix} \tag{1.25}$$

Hadamard Gate: The Hadamard gate, which is represented by H, is another important single-qubit gate. In the computational basis $\{|0\rangle, |1\rangle\}$, it is represented by the matrix

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

On the states of the computational basis, the action of H is

$$H|0\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad H|1\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \end{bmatrix}.$$

When measured in the computational basis, the Hadamard gate yields outcomes of 0 or 1 with an equal probability of $\frac{1}{2}$, indicating the creation of an equal superposition state.

The most common single-qubit gates are included in Table.(1.1) along with their circuit representation when displayed as gates in a quantum circuit and their matrix

representation in the computational basis. All other single-qubit gates can be expressed using these methods. In fact, a general (2x2) unitary matrix, the most general single-qubit operation, can be expressed (up to global phase) as

$$U(\theta,\phi,\lambda) = \begin{pmatrix} \cos\frac{\theta}{2} & e^{-i\lambda}\sin\frac{\theta}{2} \\ e^{i\phi}\sin\frac{\theta}{2} & e^{i(\phi+\lambda)}\cos\frac{\theta}{2} \end{pmatrix} \tag{1.26}$$

Additionally, some angles may be broken down into a series of rotations, such as $U(\theta,\phi,\lambda) = e^{i\alpha}R_z(\beta)R_y(\gamma)R_z(\delta)$ [5]. It is also possible to perform further decompositions into elementary rotations.

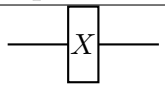
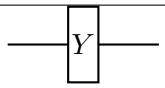
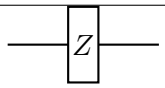
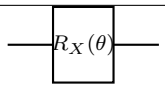
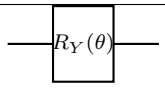
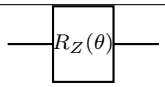
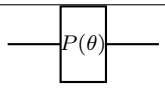
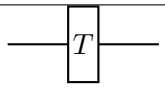
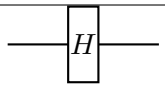
Name	**Circuit Representation**	**Matrix Representation**
Pauli-X		$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
Pauli-Y		$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$
Pauli-Z		$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
$R_X(\theta)$		$\begin{bmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix}$
$R_Y(\theta)$		$\begin{bmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix}$
$R_Z(\theta)$		$\begin{bmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{bmatrix}$
Phase Gate		$\begin{bmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{bmatrix}$
T Gate		$\begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}$
Hadamard		$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$

Table 1.1: Summary of the most important single-qubit operations. The names, abbreviations, circuit diagrams when used as gates within a quantum circuit, and matrix representations within the computational basis are given here. It is worth noting a difference between the phase gates P and R_z, though they are distinguished only by a global phase. Table from the Ref. [5].

1.5.1 Multiple Qubit Gate:

To perform a quantum computation or information processing activity, multiple qubits are typically utilized in tandem. The tensor product of the single-qubit Hilbert spaces makes the Hilbert space for a multi-qubit system. For example, a Hilbert space contains a two-qubit system.

$$\mathcal{H} = \mathcal{H}_1 \otimes \mathcal{H}_2 = (\mathbb{C}^2)^{\otimes 2}.$$

the computation is based on $\mathbb{C}^4$, which is a linear combination of the four states.

$$|\psi\rangle = \alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle, \quad \alpha, \beta, \gamma, \delta \in \mathbb{C}, \quad |\alpha|^2 + |\beta|^2 + |\gamma|^2 + |\delta|^2 = 1 \quad (1.27)$$

where the basis states $|00\rangle, |01\rangle, |10\rangle, |11\rangle$ are the outcome of considering the tensor products of the single qubits' basis states, specifically

$$|00\rangle := |0\rangle \otimes |0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\ 0 \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix},$$

$$|01\rangle = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \quad |10\rangle = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \quad |11\rangle = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}. \quad (1.28)$$

Two-qubit operations A unitary matrix in Hilbert space can describe a one-qubit operation on the state of n-qubits by $(\mathbb{C}^2)^n \times (\mathbb{C}^2)^n$. Therefore, any such matrix is a well-formed multi-qubit quantum gate. Yet, for a collection of single- and two-qubit gates to be universal for quantum computation, multi-qubit gates are commonly built from single-qubit gates rather than with random unitary matrices.

Next, let us consider some standard two-level gates. First of all, the tensor product operation can be used to compose, allowing two individual single-qubit gates to be combined to produce a resultant combined two-qubit gate on two distinct qubits. For example, the operator describes the operation of Pauli-Z gates running in parallel on a

two-qubit system [5]:

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

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.29}$$

These are quantum operations that affect two or more qubits simultaneously. These gates are required for quantum entanglement and performing operations with many qubits. Some popular multi-qubit gates include [5]:

1. **Controlled-Z gate:** In quantum computing, the Controlled-Z (CZ) gate is a two-qubit gate. It uses a pair of qubits, one of which acts as the target and the other as the control. The CZ gate applies a phase flip (relative phase) to the target qubit if the control qubit is $|1\rangle$. The CZ gate does nothing to the target qubit when the control qubit is $|0\rangle$.

   Consider a two-qubit state with the CZ gate applied to qubits that are in superposition:

   $$|\psi\rangle = u_{00}|00\rangle + u_{01}|01\rangle + u_{10}|10\rangle + u_{11}|11\rangle \tag{1.30}$$

   When we apply the CZ gate to this state, we get the following result:

   $$CZ|\psi\rangle = u_{00}|00\rangle + u_{01}|01\rangle + u_{10}|10\rangle - u_{11}|11\rangle \tag{1.31}$$

   The controlled Z gate is represented by the following matrix:

   $$\text{CZ} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \tag{1.32}$$

2. **CNOT Gate:** The two-qubit quantum register is used by the CNOT gate. One control qubit (often denoted as C) and one target qubit. When the control qubit is $|1\rangle$, the target qubit is subjected to a NOT operation. The target qubit is unaltered

otherwise. To construct quantum circuits and produce entanglement, this gate is necessary.

$$\begin{aligned} U_{\text{CNOT}}: \quad & |00\rangle \mapsto |00\rangle \\ & |01\rangle \mapsto |01\rangle \\ & |10\rangle \mapsto |11\rangle \\ & |11\rangle \mapsto |10\rangle \end{aligned}$$

The matrix representation of CNOT gate is:

$$\text{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{1.33}$$

3. **Swap Gate:** Two qubits states are switched using the swap gate. Numerous quantum algorithms can make use of it, especially those that reorganize qubit states while they are being processed. A tensor product state is affected by the SWAP gate.

$$U_{\text{SWAP}}|\psi_1, \psi_2\rangle = |\psi_2, \psi_1\rangle.$$

The following is U-SWAP's explicit form:

$$U_{\text{SWAP}} = |00\rangle\langle 00| + |01\rangle\langle 10| + |10\rangle\langle 01| + |11\rangle\langle 11|$$

The swap gate's matrix form can be shown as:

$$U_{\text{swap}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.34}$$

4. **Toffoli Gate(CCNOT):** The CCNOT gate is another name for the Toffoli gate, named after Tommaso Toffoli. Three qubits make up the Toffoli gate:one target qubit and two control qubits. It applies a NOT operation to the target qubit if both control qubits are $|1\rangle$; otherwise, it remains unchanged.

$$U_{\text{CCNOT}} = (|00\rangle\langle 00| + |01\rangle\langle 01| + |10\rangle\langle 10|) \otimes I + |11\rangle\langle 11| \otimes X$$

The computational foundation for three-qubit systems can be represented as described in [7, 5]

$$\{|000\rangle, |001\rangle, |010\rangle, |011\rangle, |100\rangle, |101\rangle, |110\rangle, |111\rangle\}.$$

The controlled-controlled-NOT (CCNOT) gate acts on these basis states:

$$\begin{aligned}
U_{\text{CCNOT}}|000\rangle &= |000\rangle, \\
U_{\text{CCNOT}}|001\rangle &= |001\rangle, \\
U_{\text{CCNOT}}|010\rangle &= |010\rangle, \\
U_{\text{CCNOT}}|011\rangle &= |011\rangle, \\
U_{\text{CCNOT}}|100\rangle &= |100\rangle, \\
U_{\text{CCNOT}}|101\rangle &= |101\rangle, \\
U_{\text{CCNOT}}|110\rangle &= |111\rangle, \\
U_{\text{CCNOT}}|111\rangle &= |110\rangle.
\end{aligned}$$

This demonstrates that the target qubit is flipped only when both control qubits are in the state ($|1\rangle$).

The Toffoli gate is depicted in matrix form as seen below:

$$CCNOT = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{1.35}$$

5. **Fredkin Gate(CSWAP):** Named for Edward Fredkin, the Fredkin gate is a three-bit gate that performs controlled swapping. It is sometimes referred to as the CSWAP or CS gate. The Fredkin gate, which is a three-qubit gate, switches the states of the second (middle) and third (bottom) qubits when the first (top) qubit is in the state $|1\rangle$. It is extensively utilized in quantum communication protocols and reversible computation. In its explicit form, it is

$$U_{\text{Fredkin}} = |0\rangle\langle 0| \otimes I_4 + |1\rangle\langle 1| \otimes U_{\text{SWAP}}$$

The CSWAP gate is represented by the following matrix:

$$CSWAP = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{1.36}$$

6. **Controlled-U gate:** A controlled-U gate is a two-qubit gate in which the first qubit serves as the control and the second as the target. The unitary operation U only affects the target qubit if the control qubit is in the state $|1\rangle$. The basis states are the following [7, 5]:

$$\begin{aligned} |00\rangle &\mapsto |00\rangle \\ |01\rangle &\mapsto |01\rangle \\ |10\rangle &\mapsto |1\rangle \otimes U|0\rangle = |1\rangle \otimes (u_{00}|0\rangle + u_{10}|1\rangle) \\ |11\rangle &\mapsto |1\rangle \otimes U|1\rangle = |1\rangle \otimes (u_{01}|0\rangle + u_{11}|1\rangle) \end{aligned}$$

The controlled-U gate's matrix representation is:

$$CU = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & u_{00} & u_{01} \\ 0 & 0 & u_{10} & u_{11} \end{bmatrix}$$

The following examples represent a small selection of quantum gates that have been developed by researchers in the field for the purpose of manipulating qubits in quantum computation. Each gate fulfills a distinct function in the execution of quantum operations and the construction of quantum algorithms, thereby contributing to the transformative capacity of quantum computing. We summarize a few popular two-qubit gates in the Table.(1.2), demonstrating their matrix representation in the computational basis as well as their circuit structure.

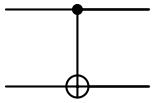
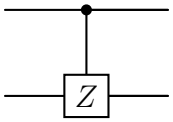
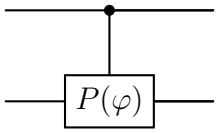
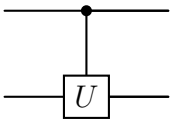
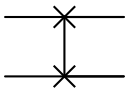
Name (abbr.)	Circuit	Matrix
Controlled-NOT (CNOT)		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$
Controlled-Z (CZ)		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$
Controlled-phase (CP)		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{i\varphi} \end{bmatrix}$
Controlled-U (CU)		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & U_{00} & U_{01} \\ 0 & 0 & U_{10} & U_{11} \end{bmatrix}$
Swap (SWAP)		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Table 1.2: Circuit symbols and matrix representations of common two-qubit gates. Table from the Ref. [5].

1.6 Measurements and Expectation values

Quantum measurement provides a mechanism for extracting classical information from quantum states. These are usually carried out to read out the outcomes of the implemented data processing task after the conclusion of a quantum computation, while they can occasionally take place throughout it, depending on the task. The probability of obtaining a measurement outcome is determined using Born's rule.

The probability that a quantum system with quantum state $|\psi\rangle = \sum_i c_i|i\rangle$ will collapse to state k depends on the computational basis.

$$p_k = |\langle k|\psi\rangle| = |c_k|^2 \tag{1.37}$$

Apart from computational quantities, some physical properties of the quantum system can be determined from a measurement. Because these properties can only exist

as real quantities (not complex), the set of physical quantities that an observer can measure, called observables, is expressed mathematically as Hermitian operators with real eigenvalues, $O = O^\dagger$. The Pauli matrices X, Y, and Z are some examples of these kinds of observables. An observable O in a quantum system with a pure state $|\psi\rangle$ has an expectation value.

$$\langle O \rangle = \langle \psi | O | \psi \rangle \tag{1.38}$$

More broadly, the following are the expected values in the density matrix formalism:

$$\langle O \rangle = \mathrm{Tr}(O\rho). \tag{1.39}$$

In the case of pure states, this formula reduces to the first one, $\mathrm{Tr}(O|\psi\rangle\langle\psi|) = \langle\psi|O|\psi\rangle$. Observables are Hermitian operators, so O admits a spectral decomposition, which is $O = \sum_i O_i |O_i\rangle\langle O_i|$. Therefore, Eq.(1.39) can also be expressed explicitly as follows:

$$\mathrm{Tr}[O\rho] = \mathrm{Tr}\left[\sum_i O_i |O_i\rangle\langle O_i|\rho\right] = \sum_i O_i \langle O_i|\rho|O_i\rangle = \sum_i O_i \rho_{ii}, \tag{1.40}$$

where the probability of measuring the state ρ in the $|O_i\rangle$ basis has diagonal elements $\rho_{ii} = \langle O_i|\rho|O_i\rangle$.

In practical quantum experiments [8, 5], estimating the expected value of an observable requires a multi-step process. The quantum state must be prepared and measured frequently, and the results must then be combined using traditional post-processing. In the context of quantum computing, one measurement act is often called a "shot". The shots are numbered, commonly denoted by M determines the statistical accuracy of the estimate, which improves with $\mathcal{O}(1/\sqrt{M})$. A summary and detailed explanation of the steps needed to estimate the expected value of an observable O on a quantum state ρ are provided in Algorithm (9).

Initially, the experimenter needs to be able to efficiently prepare the quantum state ρ multiple times or have access to many copies of it. After preparation of the quantum state, a unitary transformation rotates ρ to the eigen-basis of O. In this basis, O becomes diagonal because quantum measurements always occur in the fixed computational basis. This is typically a necessary step since all estimation processes must be re-framed using the computational foundation, which is the only way that common quantum computing gear can make projective measurements. Given that O is Hermitian, the observable is diagonalized by a unitary operator U so that $O \to O' = U^\dagger O U = \sum_i O_i |i\rangle\langle i|$. Next, one can accurately acquire the desired expectation value by rotating the quantum state

Algorithm 1 Calculate an observables expectation value

Data: Quantum state ρ, observable O, number of shots M.

Result: $\bar{O} \approx \langle O \rangle = \mathrm{Tr}[\rho O]$ with error $\mathcal{O}(1/\sqrt{M})$.

for $m = 1$ to M **do**

Prepare ρ;

Change the basis U on ρ so that $U^\dagger O U = \sum_{i=0}^{d-1} O_i \left|i\right\rangle \left\langle i\right|$ is diagonal;

A measure based on computation $\{\left|i\right\rangle\}_{i=0}^{d-1}$, find state $\left|k\right\rangle$;

The store output $r_m = O_k$;

end for

Compute Conventionally $\bar{O} := \frac{1}{M} \sum_{m=1}^{M} r_m$.

$\rho = \rho' = U^\dagger \rho U$ and measuring the diagonal, which is observable O' in the computational basis.

$$\sum_{i=0}^{d-1} O_i \left\langle i\right| U^\dagger \rho U \left|i\right\rangle = \mathrm{Tr}\left(\sum_{i=0}^{d-1} O_i \left|i\right\rangle \left\langle i\right| U^\dagger \rho U \right) = \mathrm{Tr}\left(U^\dagger O U \, U^\dagger \rho U \right) = \mathrm{Tr}(O\rho) = \langle O \rangle \tag{1.41}$$

The first term on the left defines $\mathrm{Tr}[O'\rho']$ from Eq.(1.40), which only requires measurement along the computational basis $\left|i\right\rangle$. For illustration, let us examine a single-qubit quantum state ρ and measure the predicted value of the Pauli-X operator on it. Using $Z = HXH$, it is easy to confirm that when the qubit is rotated using a Hadamard gate $\rho \to \rho' = H\rho H$ and the experimentally accessible Pauli-Z operator is measured, the desired expectation value $\langle X \rangle$ is generated. In fact,

$$\langle Z \rangle_{\rho'} = \mathrm{Tr}[Z\rho'] = \mathrm{Tr}[ZH\rho H] = \mathrm{Tr}[HZH\rho] = \mathrm{Tr}[X\rho] = \langle X \rangle_\rho. \tag{1.42}$$

We emphasize that the basis step shift is not contributing anything meaningful to the computation and is simply a practical convenience to prevent hardware limitations that allow measurement only on the computational basis.

Also, prior to measurement, the unitary operator U, diagonalizing the observable O, must be computed classically. After preparation of the quantum state ρ, the system is measured in the computational basis once the U has been applied. Measurement outcome corresponds to an eigenvalue O_k in case the state is in the eigenstate $\left|k\right\rangle$ at the time of measurement. By repeating this preparing and measuring process M times, an output sequence $\{r_1, \ldots, r_M\}$ is generated such that the result of the m-th shot is represented by $r_m \in (O_0, O_1, \ldots, O_{d-1})$. These outcomes are independent random variables, whose

statistical attributes are explained by

$$\mathbb{E}[r_m] = \mathbb{E}[O] := \langle O \rangle, \quad \mathrm{Var}[r_m] = \mathrm{Var}[O] := \langle O^2 \rangle - \langle O \rangle^2, \quad \forall m = 1, \ldots, M \tag{1.43}$$

where the expectation values and variances of the random variables r_m are calculated in accordance with Born's rule (1.37), resulting in the probability distribution of the measurement value. $\overline{O} := \sum_{m=1}^{M} r_m/M$ with variance is the sample mean. The actual expectation value $\langle O \rangle$ can be objectively estimated as follows:

$$\mathbb{E}[\overline{O}] = \frac{1}{M} \sum_{m=1}^{M} \mathbb{E}[r_m] = \langle O \rangle, \tag{1.44}$$

$$\mathrm{Var}[\overline{O}] = \frac{1}{M^2} \sum_{m=1}^{M} \mathrm{Var}[r_m] = \frac{1}{M} \mathrm{Var}[O], \tag{1.45}$$

We shifted the variance into the sum in the second equation. In addition to the fact that the r_m are independent random variables, we have done this using Eq.(1.43). When the variance of observables composed of the tensor product of Pauli matrices is bounded and unaffected by the size of the system, then

$$\mathrm{Var}[O] = \langle O^2 \rangle - \langle O \rangle^2 = [\mathrm{Tr} O^2 \rho] - [\mathrm{Tr} O \rho]^2 \in \mathcal{O}(1), \tag{1.46}$$

So, the statistical error that comes with the empirical mean $\overline{O}$ is $\mathrm{Std}[\overline{O}] \in \mathcal{O}(1/\sqrt{M})$.

Alternatively, Chebyshev's inequality can be used to estimate the expectation value with a failure probability of no more than (δ) and an accuracy of (ε).

$$P(|\overline{O} - \langle O \rangle| > \varepsilon) \leq \frac{Var[\overline{O}]}{\varepsilon^2} = \delta \rightarrow M \in \mathcal{O}\left(\frac{1}{\delta \varepsilon^2}\right), \tag{1.47}$$

It claims that with desired precision, the number of measurements, M, required to achieve a target accuracy, (ε), increases quadratically. Hoeffding's inequality [9] is one example of how other statistical inequalities can be scaled similarly.

Lastly, we note that it is often useful to measure both simple observables made up of tensor products of Pauli matrices, often called Pauli strings, in real-world applications

$$O = \sigma_1 \otimes \sigma_2 \otimes ..... \otimes \sigma_n, \tag{1.48}$$

where n is the number of qubit systems or the weighted sum of these Pauli strings.

$$O = \sum_{k=1}^{P} \gamma_k O_k = \sum_{k=1}^{P} \gamma_k \sigma_1^{(k)} \otimes \sigma_2^{(k)} \otimes ..... \otimes \sigma_n^{(k)}, \tag{1.49}$$

where the coefficients are real to guarantee Hermiticity $\gamma_k \in \mathbb{R}$ and $\sigma_i^{(k)} \in (\mathbb{I}, X, Y, Z)$ are single-qubit Pauli matrices. The standard method in this situation, which is especially

common in applications involving quantum chemistry [10], is to estimate each Pauli string independently $\langle O_k \rangle$, and then compute the results using the coefficients γ_k in a classical manner using $\langle O \rangle = \sum_k \gamma_k \langle O_k \rangle$.

Measuring a single Pauli string is easier than measuring more complicated observables with a set number of Measurement shots, and the estimate has a lower variance. Numerous methodologies have been proposed in the literature to minimize the measurement resources required to estimate expectation values, as delineated in Eq.(1.49). Most of these methods rely on a smart grouping of Pauli strings that can be measured simultaneously and commute [11]. Adaptive methods, for instance, are used to reduce the statistical variations that come with the estimation [10].

1.7 Quantum Circuit Model

As indicated previously, an ordinary quantum computation is describable via single- and two-qubit operations, so it is simple to describe graphically in terms of a string of a few circuit symbols, as in Tables (1.1) and (1.2). Gates in quantum computing are typically constructed into circuits with qubits as inputs.

Quantum circuit wires are represented by horizontal lines, and qubits move from left to right (similar to classical circuits). Gates are shown as rectangles that contain the same number of wires as the qubits they operate on. A final measurement translates the result into a list of classical bits in binary form. For controlled-U gates, it is standard practice to put a black dot on the control qubit in the line that it crosses. Below is an example of a quantum circuit:

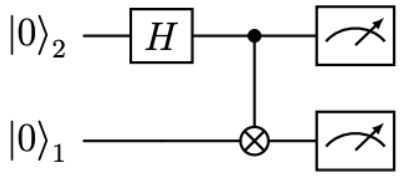


Figure 1.2: A Simple Quantum Circuit. Figure from the Ref. [12].

After applying the Hadamard gate to the first qubit, a CNOT operation is performed with the second qubit acting as the target and the first qubit acting as control. Finally,

the circuit's evolution is measured. In general, given a state $|q_1 q_2\rangle$, the left qubit q_1 is the second qubit and takes position at the top of the circuit, while the right qubit q_2 is the first qubit placed at the bottom of the circuit. The subscripts in the input qubit states indicate this convention. In this case, $|00\rangle$ represents the input quantum state.

Another simple quantum circuit that generates and measures the three-qubit GHZ state, which is a combination of $|000\rangle$ and $|111\rangle$.

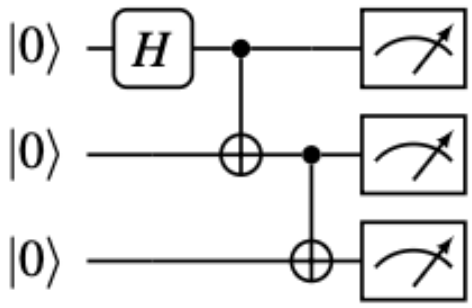


Figure 1.3: Maximally entangled GHZ state ($|000\rangle + |111\rangle/2$). Figure from the Ref. [13].

Using gates to build quantum circuits, any quantum algorithm may be defined, and this principle is central to Universal Quantum Computing. The circuit topology enables sequential computations on a small collection of qubits, modifying them using quantum features such as superposition and interference.

1.8 Real-world quantum device implementation of a quantum circuit

The quantum circuit model is a useful tool for giving a visual representation of the logical process of quantum computation. Moreover, while quantum circuits are the common format used to send commands to a genuine quantum computer for execution, they are far from accurate representations of what happens on the device.

To put a logical quantum circuit on a physical device, it needs to go through a number of classical preprocessing steps. First, choose a group of the machine's physical qubits that can connect with the two-qubit gates of the circuit that needs to be run. Second, each gate of the original circuit needs to be re-expressed in terms of the available set of physical operations on the device, referred to as native basis gates.

We list the native gates on some of the leading quantum computing hardware in Table

(1.3). They are technology-dependent to a large extent on what is realized to construct the quantum computer.

Manufacturer	Technology	Native gates	Refs.
IBM	Superconducting	$\{\sqrt{X}, X, R_Z, \text{CNOT}\}$	[14]
Google	Superconducting	$\{U(\theta, \varphi, \lambda), \text{Sycamore gate}\}$	[15], [16]
IonQ	Trapped-ions	$\{R_X, R_Y, \text{Mølmer–Sørensen}\}$	[17], [18]

Table 1.3: Some modern quantum computers have single- and two-qubit gates. The "Sycamore gate" is a two-qubit gate that looks like a mix of a controlled phase rotation and a SWAP gate. The $U(\theta, \varphi, \lambda)$ operation is the normal single-qubit rotation. The Mølmer–Sørensen gate [19] is a two-qubit gate of the form $XX(\phi) = e^{-i\phi X \otimes X/2}$ and is often used in ion-trap-based architectures. IonQ devices have single-qubit rotation gates that are variations of R_X and R_Y rotations; see [20] for a precise definition.

Transpiration or circuit compilation is the process of rewriting the circuit into a format that works for the machine. This makes a new quantum circuit that is (almost) the same as the original but is easier for the machine to run. The theoretical and mathematical underpinnings of quantum computing will be covered in the following section. However, it's crucial to first discuss the difficulties in putting quantum algorithms into practice on actual hardware. Because physical quantum circuits rely on a finite number of executions (shots), they often produce noisy outputs, making accurate computation and interpretation more difficult.

1.9 The NISQ Era of Quantum Computation

Although enormous technological progress has enabled the development of the first generations of quantum computers, we are still a long way from realizing universal fault-tolerant quantum computers, which are ideal quantum computers that can detect and correct errors in computation using a procedure known as Quantum Error Correction (QEC) [5]. Instead, the current era of quantum computing has been dubbed Noisy Intermediate-Scale Quantum [25], or NISQ for short. This term refers to near-term quantum devices, which are limited in scale due to their inability to utilize many qubits and their imperfections, which make them highly susceptible to noise.

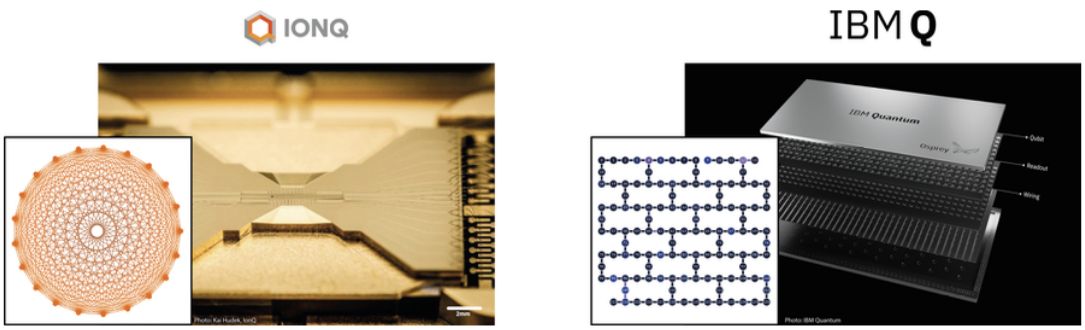


Figure 1.4: Superconducting circuits (IBM Quantum) [21] and trapped ions (IonQ) [22] are two examples of quantum computers that are already in use and use different technologies. The panels next to the devices show examples of how qubits are usually connected in the technologies that go with them. For example, IBM's Washington device has n=127 qubits and connects in a grid-like way [23], while the Aria device has n=21 qubits and connects all-to-all [24]. Figure from Ref. [23, 24].

Quantum computing solutions [26], [27] are being developed on several experimental platforms, including superconducting circuits [15], [21], trapped ions [22], [28], neutral atoms [29], [30], photonic chips [31], [32], and others. Two examples of currently available quantum computers using trapped ions and superconducting circuits, possibly the most promising technologies for building quantum computers in the NISQ era, are shown in Fig.(1.4). The figure shows the arrangement of the qubits on these chips graphically as well as the actual Quantum Processing Units (QPUs). Even though the rate of development of these machines points to a quantum version of the classical Moore's law, NISQ devices still have several significant limitations, which we list below:

Limitations of NISQ Devices

As previously stated, the present NISQ devices have certain restrictions, specifically:

- **Size Limitations:** There are usually only a few qubits in NISQ-era quantum devices. At the present time, they can have anywhere from a dozen to several hundred qubits, and generally up to 1000 qubits. The complexity of the problems they can tackle is limited by their comparatively small size. The fact that the first-generation (QPUs) Quantum Processing Units that are anticipated to attain quantum advantage are still years away suggests that the current scale is insufficient to fully fulfill the potential of quantum computing.

- **Scalability Challenges:** In order to gain a clear quantum edge over traditional simulation, NISQ devices must be scaled efficiently. For instance, about 20 million physical qubits or various logical qubits are needed to construct Shor's algorithm, a crucial example of quantum advantage. A major challenge is creating such large-scale qubit systems while preserving quantum coherence and functionality, which is far from the present NISQ devices. To overcome this constraint in the upcoming decades, new experimental and algorithmic discoveries will probably be required.

- **Connectivity Constraints:** In a NISQ device, the hardware configuration determines how qubits are coupled. It can be challenging to provide complete connectivity between qubits in some implementations. For example, it might not always be feasible to create 2-qubit gates between random qubits. While all qubits in trapped ion quantum computers can communicate with one another, qubits in other devices, such as superconducting ones with a Qubit grid in two dimensions, can only communicate with their near neighbors, which are typically two or three qubits. The kinds of algorithms that these devices can efficiently run are constrained by their poor connectivity.

Due to noise and decoherence affecting their qubits, NISQ devices have limited coherence times and imprecise operations. The complexity and depth of quantum circuits that can be reliably implemented are limited by this inherent noise. Not with standing these limitations, NISQ devices provide a valuable platform for researching quantum algorithms and applications that may be better than classical approaches for workloads.

Applications

Researchers are looking into a range of NISQ device applications in several domains:

- **Quantum Simulation:**

  The ability of NISQ devices to simulate quantum many-body systems facilitates the investigation of complex materials and physical phenomena.

- **Quantum Chemistry:**

  Molecular energies can be calculated using algorithms like the VQE (Variational Quantum Eigen) solver, which may provide information on chemical reactions and characteristics.

- **Optimization problems:**

  The Quantum Approximate Optimization Algorithm (QAOA) can be used to solve combinatorial optimization problems, which are common in many fields.

- **Machine Learning:**

  To improve data analysis and pattern identification work, quantum machine learning algorithms are being investigated.

Since each approach has benefits and drawbacks, it is still unclear which will be the most efficient way to create scalable and fault-tolerant quantum computers. Near-term technologies offer a new paradigm for computation as well as new tools to explore the boundaries of quantum mechanics. Furthermore, they are a crucial stage in the large-scale development of universal quantum computers. In the following chapter, we discuss the potential applications of NISQ devices and describe variational quantum algorithms, a kind of quantum algorithm designed especially for modern quantum computing devices.

CHAPTER 2

AN INTRODUCTION TO VARIATIONAL QUANTUM ALGORITHMS AND THEIR APPLICATIONS

2.1 Variational Quantum Algorithm

Variational quantum algorithms (VQAs) are the most promising way to take advantage of current quantum computing platforms before standard quantum algorithms, like Shor's factoring, can be applied at scale. This is predicated on the idea that the non-error-corrected regime can already yield a significant quantum advantage. Furthermore, the study of variational quantum algorithms is an interesting topic in and of itself, independent of any quantum advantage. This is because it offers compelling use cases that encourage the development of new hardware and software solutions for quantum computing and because it proposes a new paradigm of hybrid computation, which combines classical and quantum resources to complete a task.

The concept behind variational quantum algorithms is to get around the size and noise limitations of NISQ devices by using them sparingly and only for the most essential tasks, while outsourcing the rest to a classical computer. This idea then identifies a class of hybrid quantum-classical algorithms that combine quantum and classical computational resources to solve a task. The second component of variational algorithms, which gives them their name, also very similar to the the most popular field of machine learning.

Variational quantum algorithms are optimization based processes that solve problems by first encoding the solution as the minimum of a cost function that is properly defined by $C(\boldsymbol{\theta})$, which depends on some tunable parameters $\boldsymbol{\theta}$. Afterwards, using gradient-based techniques, these parameters are iteratively varied to find the function's minimum and, consequently, the solution. The versatility and power of learning-based procedures have been demonstrated by the amazing results that deep learning has produced in recent years [33, 34, 35]. To overcome the severe limitations imposed by existing NISQ hardware, freely adjustable parameters are introduced into a quantum computation through variational quantum algorithms.

2.2 Foundational Concepts

One of the primary benefits of VQAs is that they offer a comprehensive framework for resolving a range of problems, starting with parameterized quantum circuits that are run on a Quantum Processing Unit (QPU) and concluding with parameter optimization using a Classical Processing Unit (CPU).

The VQA's working principle is illustrated in Fig.(2.1); the algorithm consists of five steps:

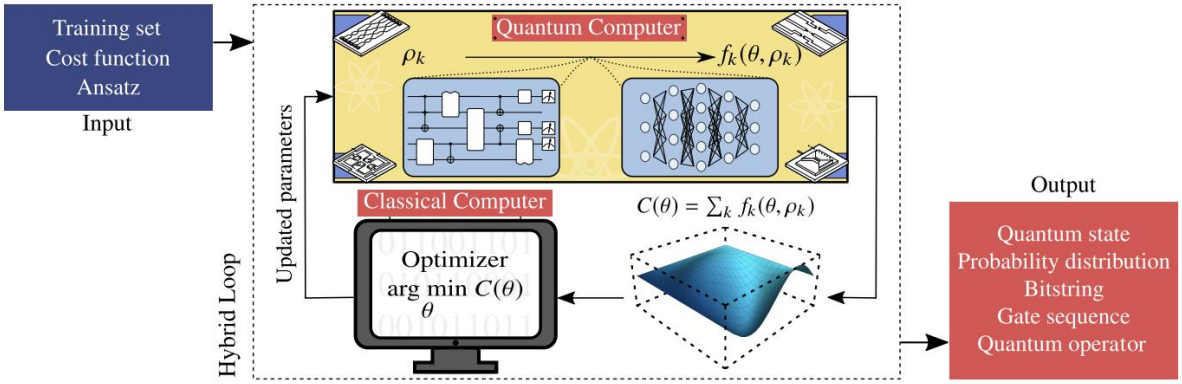


Figure 2.1: A variational quantum algorithm's (VQA) schematic diagram. Figure from Ref. [36].

- **Step 1:** Expressing the problem in terms of a cost function, $C(\boldsymbol{\theta})$ re-frames the computational work as an optimization problem, and when $C(\boldsymbol{\theta})$ hits its global minimum, the desired solution is achieved. This method employs optimization techniques to methodically navigate the parameter space and find the ideal configuration that meets the computational objective.

- **Step 2:** When theta $\boldsymbol{\theta}$ is optimized to reduce the cost function, make sure the parameterized quantum circuit ansatz is designed to become a quantum model that can resolve the goal problem.
- **Step 3:** Following the computation of the cost function $C(\boldsymbol{\theta})$ and taking the measurements of the output quantum state, the measurement data should be extracted.
- **Step 4:** Utilizing the classical parameter optimizers to update the $\boldsymbol{\theta}$.
- **Step 5:** Check if the cost function has reached the convergence, stop the procedure. Alternatively it is possible to fix a maximum number of iterations before the procedure finishes.

When constructing a VQA, one has to think about how to express an ansatz and how to express the cost function to express the computational task into an optimization problem, as shown in Fig.(2.1). Furthermore, the algorithm's accuracy and efficiency may be impacted by the classical optimizer that is selected. We go over these fundamental ideas in VQAs in the sections that follow.

2.3 Cost Function

The development of an appropriate cost function is one of the most crucial phases in Variational Quantum Algorithms (VQAs). The cost function, like its classical counterpart, converts the learnable parameters of a quantum circuit to real numbers, essentially establishing the cost landscape over which classical optimization methods seek global minima. The following is the typical representation of the cost function:

$$C(\boldsymbol{\theta}) = \sum_j f_j \operatorname{Tr}\left[H_j\, U(\boldsymbol{\theta})\, \rho\, U^\dagger(\boldsymbol{\theta})\right] = \sum_j f_j \langle H_j \rangle, \tag{2.1}$$

here, ρ represents the initial density matrix, $U(\boldsymbol{\theta})$ represents the parametrized quantum circuit, and H_j represents a set of observables. To define the cost function, the classical coefficients f_j are used to combine the related expectation values [37, 36].

2.4 Parameterized quantum Ansatz:

In variational quantum algorithms, as in ordinary parametric models, the functional form of the model needs to be specified a prior, which is accomplished by taking into account

a particular parameterized quantum circuit $U(\boldsymbol{\theta})$. Such a selection is known as an ansatz circuit, and although there isn't a single, well-defined method for creating one, there are certain general guidelines that can be applied to find suitable candidates.

The ansatz explains how to optimize the parameter $\boldsymbol{\theta}$ to minimize the cost function, making it a crucial component of variational quantum circuits. Utilizing the unitary operator $U(\boldsymbol{\theta})$ to the quantum circuit's input state usually encodes the parameter $\boldsymbol{\theta}$.

A common way to express this unitary operator is as the product of unitaries applied one after the other. The purpose of $U(\boldsymbol{\theta})$ is to approximate the solution, much like the trial function in the variational technique of classical quantum mechanics. It is important to understand that, although any choice of $U(\boldsymbol{\theta})$ can potentially produce results, the precision of the solution can be affected by the wide range of results that can be obtained depending on the ansatz used. Generally, the $U(\boldsymbol{\theta})$ can be written like this:

$$U(\boldsymbol{\theta}) = \sum_{\ell=1}^{L} V_\ell U_\ell(\theta_\ell) = V_L U_L(\theta_L) \dots V_1 U_1(\theta_1) \tag{2.2}$$

where $U_l(\theta_l) = \exp(-i\theta_l H_l)$, L is the circuit's layer number, V_l is the entangling gate that is independent of any θ_l, and U_l is some hermitian operator. The formula for $U(\boldsymbol{\theta})$ in Eq.(2.2) is a general structure that can be changed in several forms depending on the problem. The performance of VQA is significantly impacted by the ansatz selection. Ansatz influences how close the final state is to the ideal state for resolving the issue, but it also requires consideration of the quantum hardware used for the VQA. Building certain ansatz gates from local gates can be costly, and deeper circuits are sensitive to errors. Thus, there are two types of ansatz: a hardware-efficient ansatz and a problem-inspired ansatz.

2.4.1 Problem-based ansatz:

- **Ansatz of unitary coupled clusters:** In quantum chemistry, the main application of the Unitary Coupled Cluster (UCC) ansatz [38] is to supplement the Hartree-Fock approximation with quantum correlations. Using the UCC ansatz, Peruzzo et al. [37] implemented the Variational Quantum Eigensolver (VQE). A parameterized cluster operator $T(\boldsymbol{\theta})$ makes up the UCC ansatz, which operates on the Hartree-Fock ground state $|\psi_{HF}\rangle$.

$$|\psi(\boldsymbol{\theta})\rangle = e^{T(\boldsymbol{\theta}) - T(\boldsymbol{\theta})^\dagger} |\psi_{HF}\rangle \tag{2.3}$$

- **Ansatz for Quantum alternating operator:** The Quantum Approximate Optimization Algorithm (QAOA) [39] alternates between mixing Hamiltonians and cost functions to handle combinatorial optimization. The (QAOA) Quantum Alternating Operator Ansatz, developed by Hadfield et al. [40], extended this by extending unitary families beyond specified local Hamiltonians. This ansatz can, compared to the original one, describe a larger, potentially more beneficial set of states. As Lloyd et al. [41] formulate it, such an ansatz has a generic set of gates and is able to do generalized quantum computation for one-dimensional nearest-neighbor local cost Hamiltonians with ZZ terms.

  Morales et al. [42] used families of ansatz based on families of graphs and hypergraphs to generalize the universality of this set further. The quantum alternating operator ansatz has the following form:

$$U(\boldsymbol{\beta}, \boldsymbol{\gamma}) = \prod_{\ell=1}^{L} e^{-i\boldsymbol{\beta}_\ell H_M} e^{-i\boldsymbol{\gamma}_\ell H_P}, \tag{2.4}$$

  where $(\boldsymbol{\beta}, \boldsymbol{\gamma})$ are optimized parameters controlling the transition to the target ground state, L is the circuit depth, H_M is the mixer Hamiltonian, and H_P is the problem Hamiltonian encoding the solution.

- **Variational Hamiltonian Ansatz:** To determine a Hamiltonian's ground state, which is represented as $H = \sum_i h_i$, with h_i being Hermitian operators, Wecker et al. [43] devised the Variational Hamiltonian Ansatz (VHA), which was motivated by adiabatic state preparation. Each Trotter step in this technique's Trotterized adiabatic process functions as a variational ansatz. As a result, the Unitary Operator is provided by:

$$U_{VHA} = \prod_i e^{(i\boldsymbol{\theta} h_i)} \tag{2.5}$$

  Experiments show that VHA is superior to a particular version of the unitary coupled cluster (UCC) ansatz for modeling systems of tightly coupled quantum chemistry.

2.4.2 Effective hardware ansatz

Compared to problem-inspired ansatz, the hardware efficient ansatz [44] is based on coherence time, limited gate fidelity, limited gate set, and qubit connectivity, all characteristics of quantum hardware [45] to guarantee effective implementation on NISQ

devices. These kinds of ansatz are characterized by the usage of a certain qubit connection topology and a restricted number of quantum gates.

The circuit is made up of blocks that have entanglement gates and single qubit gates applied to several qubits in parallel. Gate sets typically contain two qubit entanglement gates and up to three single qubit gates. Ansatz consists of several layers, each of which is called a layer. An L-layer hardware-efficient ansatz looks like this:

$$U(\boldsymbol{\theta}) = \prod_{k=1}^{L} U_k(\boldsymbol{\theta}) V_k \tag{2.6}$$

where $U_k(\theta_k) = e^{-i\theta_k H_k}$ is obtained from the Hermitian operator's unitary H_k. where the variational parameter is θ_k. A Non-parametrized quantum gate is denoted by V_k. The ansatz's capacity to cover the whole Hilbert space and its rate of convergence are influenced by the choice, connection, and arrangement of its gates. The literature has examined a number of relevant characteristics of hardware efficient ansatz, such as trainability [46, 47], performance on benchmark problems [48], entangling capability [49], and expressibility [50].

2.5 Parameter optimization

The cost function's value is then determined by quantum measurement operations that yield the classical results of the output quantum state of ansatz under the parameter $\boldsymbol{\theta}$ after the cost function and ansatz have been determined. Using a parameter optimizer to update the parameter $\boldsymbol{\theta}$ and determine the local optimal value of the specified cost function is the optimization principle of VQA. Thus, the effectiveness of VQA is dependent on the efficiency and dependability of the optimization approach used. The optimization methods are divided into two major groups: Gradient-based and Gradient-free methods

2.5.1 Gradient-based techniques

One of the widely used and natural optimization strategies is the gradient descent method [51], and it has been extensively researched in classical machine learning. This strategy iteratively updates parameters by shifting in the direction of the function's negative gradient at each step.

We rely on Stochastic Gradient Descent (SGD), which uses a portion of the parameter space to approximate the true gradient because accurate gradients are frequently

unavailable. Instead of computing the entire gradient, SGD estimates it using expectation values derived from a finite number of samples. Generally, gradient descent techniques work by iteratively changing parameters in the negative of gradient direction. If the gradient is $\nabla f(x_k) \neq 0$, the search direction is denoted as $d_k = -\nabla f(x_k)$, which follows the anti-gradient direction at step k. This method is frequently employed in optimization, where the gradient $\nabla_{\boldsymbol{\theta}} C(\boldsymbol{\theta})$ of a cost function $C(\boldsymbol{\theta})$ with respect to its parameter $(\boldsymbol{\theta})$ is calculated.

The following equation demonstrates how gradient descent is used iteratively to update the parameter so that it moves in the direction indicated by the gradient:

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \eta \, \nabla_{\boldsymbol{\theta}} C(\boldsymbol{\theta})|_{\boldsymbol{\theta}^{(t)}} \tag{2.7}$$

where $0 < \eta << 1$ is a learning rate hyperparameter that modifies the algorithm's step size. This update rule will propose a new parameter value that corresponds to a lower cost if the learning rate is low enough.

Taylor demonstrates this by enlarging the cost at step $t+1$ around the parameters of the preceding step t, as

$$\begin{aligned} C\left(\boldsymbol{\theta}^{(t+1)}\right) &= C\left(\boldsymbol{\theta}^{(t)} - \eta \nabla_{\boldsymbol{\theta}} C(\boldsymbol{\theta})\Big|_{\boldsymbol{\theta}^{(t)}}\right) \\ &\simeq C\left(\boldsymbol{\theta}^{(t)}\right) + \nabla_{\boldsymbol{\theta}} C\left(\boldsymbol{\theta}^{(t)}\right)\left[-\eta \nabla_{\boldsymbol{\theta}} C\left(\boldsymbol{\theta}^{(t)}\right)\right] + \mathcal{O}(\eta^2) \\ &= C\left(\boldsymbol{\theta}^{(t)}\right) - \eta \left|\nabla_{\boldsymbol{\theta}} C\left(\boldsymbol{\theta}^{(t)}\right)\right|_2^2 \\ &\leq C\left(\boldsymbol{\theta}^{(t)}\right), \end{aligned}$$

here $||.||_2$ represents a vector's 2-norm (also known as its Euclidean norm), and the final inequality results from the fact that both the gradient norm and the learning rate are positive values.

The basic gradient descent rule in Eq. (2.7) can be generalized and modified by using more information about second-order derivatives to produce complex methods like BFGS [52], which are often used for local optimization. In addition to second-order algorithms, adaptive learning rate methods like AdaGrad, RMSProp, and ADAM [53, 54] dynamically modify step size to better navigate complex environments. These techniques improve robustness and efficiency in a variety of situations.

Other approaches, such as conjugate and accelerated gradient descent, improve convergence and deal with non-convexity. These advancements demonstrate the continuous improvement of gradient-based optimization. For now, we focus on finding the

gradient of the cost function for each $\boldsymbol{\theta}_i$. Now, we discuss the most significant approach, which is the basis of our analysis.

- **Parameter-Shift-Rule**

  The parameter shift rule is applicable to both hardware and simulators, was initially introduced by Schuld et al. in [55]. For gradient-based methods, the target function's first partial derivatives must be obtainable; these can be quantitatively estimated using the central finite-difference formula.

$$\partial_i C = \frac{C(\boldsymbol{\theta} + \varepsilon \boldsymbol{e_i}) - C(\boldsymbol{\theta} - \varepsilon \boldsymbol{e_i})}{2\varepsilon}, \quad 0 < \varepsilon \ll 1 \tag{2.8}$$

  where the unit vector $\boldsymbol{e_i} = (0, ...1, ...0)$ has zero entries, with the exception of a one at the i-th point. when $\varepsilon \to 0$ is the limit of vanishing displacement, the equality sign is recovered. As it turns out, derivatives of parameterized quantum circuits follow a similar but precise formula, which is now known as the parameter-shift rule [55, 47, 56, 57, 58]. Using the derivation provided in ref. [56], we demonstrate this formula, which is a direct result of the rotation-like nature of the parameterized gates employed in variational circuits.

  In variational quantum circuits, common parameterized operations include rotation like gates of the form:

$$V_j(\theta_j) = e^{-i\theta_j H_j/2} \tag{2.9}$$

  where H_j represents the Hermitian generator of the gate and θ_j is a variational angle. It is possible to recast the exponential in trigonometric form as follows when H_j is involuntary ($H_j^2 = \mathbb{I}$) [5],

$$V_j(\theta_j) = \cos\left(\frac{\theta_j}{2}\right) \mathbb{I} - i \sin\left(\frac{\theta_j}{2}\right) H_j, \tag{2.10}$$

Note that a variety of operations, such as rotations using a single qubit or, more generally, any rotation produced by tensor products of Pauli matrices, can be described in this manner.

Consider the cost function to be differentiated as $C(\boldsymbol{\theta}) = \mathrm{Tr}\left[O\, U(\boldsymbol{\theta})\, \rho\, U(\boldsymbol{\theta})^\dagger\right]$, and the parameterized gate as $V_j(\theta_j)$, which depends on the variable θ_j for which the derivative is to be computed. Examine the circuit $U(\boldsymbol{\theta})$, which is a bi-partition of all the gates acting

after (U_A) and before (U_B). In this circuit, the operation of interest $V_j(\theta_j)$ takes place [1]

$$U(\boldsymbol{\theta}) = U_A V_j(\theta_j) U_B \tag{2.11}$$

To make the notation simpler, U_B and U_A suppress the dependence on other variational parameters. Next, the cost function can be written as

$$\begin{aligned} C(\boldsymbol{\theta}) &= \mathrm{Tr}\Big[O\, U_A\, V_j(\theta_j)\, U_B\, \rho\, U_B^\dagger\, V_j^\dagger(\theta_j)\, U_A^\dagger \Big] \\ &= \mathrm{Tr}\Big[O_A\, V_j(\theta_j)\, \rho_B\, V_j(\theta_j)^\dagger) \Big] \end{aligned} \tag{2.12}$$

where $O_A = U_A^\dagger O U_A$ and $\rho_B = U_B \rho U_B^\dagger$, and we separated the cost's dependence on the relevant variable $\boldsymbol{\theta}_j$. When the above expression is substituted with the trigonometric formula (2.10), one gets

$$\begin{aligned} C(\boldsymbol{\theta}) &= \mathrm{Tr}\left[O_A \left(\cos\left(\frac{\theta_j}{2}\right) \mathbb{I} - i \sin\left(\frac{\theta_j}{2}\right) H_j \right) \rho_B \left(\cos\left(\frac{\theta_j}{2}\right) \mathbb{I} + i \sin\left(\frac{\theta_j}{2}\right) H_j^\dagger \right) \right] \\ &= \mathrm{Tr}\left[\cos^2\left(\frac{\theta_j}{2}\right) O_A \rho_B + i \sin\frac{\theta_j}{2} \cos\frac{\theta_j}{2} \left(O_A \rho_B H_j^\dagger - O_A H_j \rho_B \right) + \sin^2\frac{\theta_j}{2} O_A H_j \rho_B H_j^\dagger \right] \\ &= \frac{1+\cos\theta_j}{2} \mathrm{Tr}[O_A \rho_B] + \frac{1-\cos\theta_j}{2} \mathrm{Tr}[O_A H_j \rho_B H_j^\dagger] + \frac{i}{2} \sin\theta_j\, \mathrm{Tr}\Big[O_A \big(\rho_B H_j^\dagger - H_j \rho_B \big) \Big] \\ &= C_0 + C_1 \cos(\theta_j) + C_2 \sin(\theta_j) \end{aligned} \tag{2.13}$$

where the real integers C_0, C_1, and C_2 are independent of the parameter θ_j, which is defined as

$$\begin{aligned} C_0 &= \frac{\mathrm{Tr}\left[O_A \left(\rho_B + H_j \rho_B H_j^\dagger \right) \right]}{2}, \\ C_1 &= \frac{\mathrm{Tr}\left[O_A \left(\rho_B - H_j \rho_B H_j^\dagger \right) \right]}{2}, \\ C_2 &= \frac{\mathrm{Tr}\left[O_A \left(\rho_B - H_j \rho_B H_j^\dagger \right) \right]}{2}. \end{aligned} \tag{2.14}$$

The cost function is just a trigonometric polynomial for each variational parameter, as demonstrated by equation (2.13), which makes it interesting.

The sine and cosine function derivatives are then calculated using the following identities.

$$\begin{aligned} \frac{d\cos x}{dx} &= \frac{\cos(x+s) - \cos(x-s)}{2\sin s}, \\ \frac{d\sin x}{dx} &= \frac{\sin(x+s) - \sin(x-s)}{2\sin s} \end{aligned} \qquad \forall s \neq m\pi,\ m \in \mathbb{Z} \tag{2.15}$$

[1] If the same parameter appears many times in the circuit, the entire derivative can be calculated by separately shifting each parameterized gate and then adding the resulting contributions [55]

The derivative of the cost function with respect to the (θ_j) in Eq.(2.13) can also be expressed as:

$$\begin{aligned}\frac{\partial C(\boldsymbol{\theta})}{\partial \theta_j} &= C_1 \frac{d\cos\theta_j}{d\theta_j} + C_2 \frac{d\sin\theta_j}{d\theta_j} \\ &= \frac{C_0 + C_1\cos(\theta_j + s) + C_2\sin(\theta_j + s)}{2\sin s} - \frac{C_0 + C_1\cos(\theta_j - s) + C_2\sin(\theta_j - s)}{2\sin s} \\ &= \frac{1}{2\sin s}\left[C(\boldsymbol{\theta} + s\boldsymbol{e}_j) - C(\boldsymbol{\theta} - s\boldsymbol{e}_j)\right]\end{aligned} \tag{2.16}$$

where the variational parameter is appropriately redefined in the final line to acknowledge that both numerators are instances of the cost function in the expression (2.13). The standard formulation of the parameter-shift rule [55, 59, 57] is ultimately reached by putting $s = \pi/2$ in the above expression.

$$\frac{\partial C(\boldsymbol{\theta})}{\partial \theta_j} = \frac{1}{2}\left[C(\boldsymbol{\theta} + \frac{\pi}{2}\boldsymbol{e}_j) - C(\boldsymbol{\theta} - \frac{\pi}{2}\boldsymbol{e}_j)\right], \tag{2.17}$$

The parameter-shift rule (2.17) is not the finite-difference rule (2.8). Despite their apparent similarities, they are completely different with the main difference being that the former is a precise relationship between the derivative of a function and its value at certain places, whilst the latter is merely an approximation.

Generally, a parameter-shift rule (2.17) can be constructed if the parameterized operation is of the form (2.9) and the rotation generator has exactly two distinct eigenvalues, which relates to the requirement that the generator be involutory [60, 55]. If the above conditions do not hold, then one can make use of generalizations of the parameter-shift rules to address the challenge, e.g., decomposing the parameterized gate into a product of rotation-like operations [60] or exponentiating the generator of the unitary evolution in the Pauli basis and approximating the gradient of each element through a stochastic process [61]. Through the evaluation of two circuit instances per parameter, the parameter shift rule provides a realistic method for estimating gradients of variational quantum circuits using quantum hardware, making it a useful tool.

As shown in Eq.(2.7), a single step of the gradient descent method requires the measurement of $2p$ circuits, where p represents the number of parameters $\boldsymbol{\theta} \in \mathbb{R}^p$. Even if this scales linearly, for near-term devices, the overall cost of circuit execution rises significantly, particularly when high shot counts and several iterations are required. For instance, using p parameters to optimize across T steps and M shots results in an $\mathcal{O}(pTM)$ cost.

Classical machine learning algorithms can calculate gradients more efficiently in a

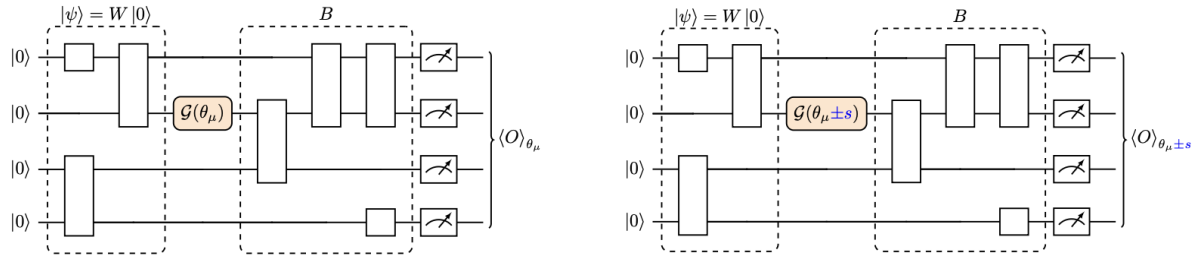


Figure 2.2: Using diagrams, we apply the parameter-shift rule to calculate the derivative of cost function. By comparing the cost function on the left diagram with the one on the right, we can determine the gradient of the cost function with precision. Figure from Ref. [55, 62].

single step using automatic differentiation and backpropagation. This involves storing intermediate values during calculation and merging them at the end using the chain rule to calculate derivatives for constructed functions. Backpropagation cannot be applied to variational quantum algorithms because measuring intermediate states disturbs the system. Finally, an ancilla-based measurement method like a Hadamard test can be used to find the gradients of parameterized quantum circuits [55, 62].

2.6 Higher order derivatives

We can examine higher-order derivatives of the cost function for a wide range of quantum circuits by expanding the parameter shift rule [56]. Specifically, by applying different shifts to the circuits, we can find the higher-order derivatives of the cost function analytically. As an illustration, the second mixed derivatives read

$$\begin{aligned}\frac{\partial^2 C(\boldsymbol{\theta})}{\partial\theta_j\partial\theta_i} &= \frac{1}{2}\left[\frac{\partial}{\partial\theta_j}C\left(\boldsymbol{\theta}+\frac{\pi}{2}\boldsymbol{e}_i\right)-\frac{\partial}{\partial\theta_j}C\left(\boldsymbol{\theta}-\frac{\pi}{2}\boldsymbol{e}_i\right)\right]\\ &= \frac{1}{4}\left[C\left(\boldsymbol{\theta}+\frac{\pi}{2}\boldsymbol{e}_i+\frac{\pi}{2}\boldsymbol{e}_j\right)-C\left(\boldsymbol{\theta}+\frac{\pi}{2}\boldsymbol{e}_i-\frac{\pi}{2}\boldsymbol{e}_j\right)\right.\\ &\quad \left.-C\left(\boldsymbol{\theta}-\frac{\pi}{2}\boldsymbol{e}_i+\frac{\pi}{2}\boldsymbol{e}_j\right)+C\left(\boldsymbol{\theta}-\frac{\pi}{2}\boldsymbol{e}_i-\frac{\pi}{2}\boldsymbol{e}_j\right)\right]\end{aligned} \tag{2.18}$$

The above expression can be computed by using the application of parameter shift rule as described in [56]:

$$\frac{\partial^2 C(\boldsymbol{\theta})}{\partial\theta_j\partial\theta_i} \approx \frac{\pi^2}{16s^2}\left[C(\boldsymbol{\theta}+s\boldsymbol{e}_i+s\boldsymbol{e}_j)-C(\boldsymbol{\theta}+s\boldsymbol{e}_i-s\boldsymbol{e}_j)-C(\boldsymbol{\theta}-s\boldsymbol{e}_i+s\boldsymbol{e}_j)+C(\boldsymbol{\theta}-s\boldsymbol{e}_i-s\boldsymbol{e}_j)\right] \tag{2.19}$$

Now use, $s = \frac{\pi}{2}$ or $s^2 = \frac{\pi^2}{4}$, we get:

$$\begin{aligned}\frac{\partial^2 C(\boldsymbol{\theta})}{\partial\theta_j\partial\theta_i} \approx \frac{1}{4}\big[C\left(\boldsymbol{\theta}+\frac{\pi}{2}\boldsymbol{e}_i+\frac{\pi}{2}\boldsymbol{e}_j\right) - C\left(\boldsymbol{\theta}+\frac{\pi}{2}\boldsymbol{e}_i-\frac{\pi}{2}\boldsymbol{e}_j\right)\\ - C\left(\boldsymbol{\theta}-\frac{\pi}{2}\boldsymbol{e}_i+\frac{\pi}{2}\boldsymbol{e}_j\right) + C\left(\boldsymbol{\theta}-\frac{\pi}{2}\boldsymbol{e}_i-\frac{\pi}{2}\boldsymbol{e}_j\right)\big]\end{aligned} \tag{2.20}$$

We may choose any value for s, but for simplicity and efficiency, $s = \frac{\pi}{2}$ and $s = \frac{\pi}{4}$ are two important options. When the shift parameter is set to $s = \frac{\pi}{4}$, the second derivative of the cost function (Hessian Matrix) becomes much simpler in the special case of $i = j$.

$$\frac{\partial^2 C(\boldsymbol{\theta})}{\partial\theta_i^2} = \frac{1}{2}\left[C\left(\boldsymbol{\theta}+\tfrac{\pi}{2}\boldsymbol{e}_i\right) - 2C\left(\boldsymbol{\theta}\right) + C\left(\boldsymbol{\theta}-\tfrac{\pi}{2}\boldsymbol{e}_i\right)\right] \tag{2.21}$$

Conversely, the formula for $\frac{\partial^2 C(\boldsymbol{\theta})}{\partial\theta_j\partial\theta_i}$ with $i = j$ for $s = \frac{\pi}{2}$ is as follows:

$$\frac{\partial^2 C(\boldsymbol{\theta})}{\partial\theta_i^2} = \frac{1}{2}\left[C\left(\boldsymbol{\theta}+\pi\boldsymbol{e}_i\right) - C(\boldsymbol{\theta})\right] \tag{2.22}$$

Two main optimization techniques are presented in the next section. They are distinguished by advanced iteration rules that contain details about the second-order derivative, particularly the cost function's convexity..

2.6.1 Newton optimizer

Newton's approach is a second-order optimization procedure [63, 56] that determines the best course of action for each iteration based on the Hessian matrix (second order of cost function) being optimized.

Unlike gradient descent and RMSProp, which only employ first-order information (the gradient), Newton's technique takes into account second-order information (the Hessian) on the shape of the function. The Newton's method update rule is as follows:

$$\boldsymbol{\theta}^{(t)} = \boldsymbol{\theta}^{(t-1)} - \eta\left[HC\big(\boldsymbol{\theta}^{(t-1)}\big)\right]^{-1}\nabla C(\boldsymbol{\theta}^{(t-1)}). \tag{2.23}$$

The Hessian matrix's inverse is represented by $[HC(\boldsymbol{\theta}^{(t-1)})]^{-1}$, which may be approximated using the second-order parameter-shift algorithm.

An approximation technique called the Diagonal Newton optimizer [64] is suggested in order to reduce the computing expense. It lowers the computing complexity by roughly representing the entire Hessian matrix as a diagonal matrix.

2.6.2 Quantum Natural Gradient Optimizer

Inspired by classical natural gradient methods [65], Quantum Natural Gradient Descent (QNGD) is a quantum optimization technique that consider the underlying geometry

of the quantum state space in order to improve convergence. In QNGD, the Euclidean metric is replaced by the Fubini-Study metric tensor [66], which describes the geometry of quantum states and functions as a quantum counterpart of the Fisher information matrix. The classical Newton method is comparable to the QNGD update rule, but it uses the Fubini-Study metric tensor in place of the Hessian. The rule for updating parameters is given by

$$\boldsymbol{\theta}^{(t)} = \boldsymbol{\theta}^{(t-1)} - \eta \left[F\left(\boldsymbol{\theta}^{(t-1)}\right)\right]^{-1} \nabla C(\boldsymbol{\theta}^{(t-1)}). \tag{2.24}$$

where η is the adaptive learning rate, $\eta[F(\boldsymbol{\theta}^{(t-1)}]^{-1}$ is the Fubini-Study metric tensor evaluated at the current parameters, and λ is a small regularization hyperparameter to ensure the invertibility of the tensor. By considering the geometric structure of the quantum state space, QNGD can provide more efficient and stable convergence during training of quantum machine learning models.

2.7 Gradient Free Methods

A set of algorithms that do not require knowledge of the cost function's derivatives are known as gradient-free optimization approaches [67].

In quantum computing, the Modified Powell's Method [68], Nelder-Mead [69], and COBYLA [70] (Linear Approximation for Constrained Optimization) are some of the main gradient-free techniques. The Nelder-Mead simplex method was created by Nelder and Mead. [69], is a derivative-free optimization strategy that evaluates the objective function at a simplex vertices and iteratively moves away from the vertex with the lowest value. This adaptive process constantly reshapes the simplex to better match the underlying response surface.

In quantum computing, cost functions are typically evaluated by measuring appropriate observables on the quantum system. However, noise introduces significant errors into this analysis. The scalability of such algorithms is constrained by the increasing precision needed to distinguish cost function values as their differences get smaller. When this precision is not achieved, optimization decisions become effectively random, resulting in behaviors similar to random walks. Furthermore, because cost function differences can be exponentially suppressed, obtaining reliable estimates necessitates an exponential number of measurements, limiting the potential for quantum speedup. Arrasmith et al. [67] showed analytically that gradient-free optimizers are susceptible to Barren Plateaus, in which the variance in cost function differences vanishes exponentially.

Because Nelder-Mead and other optimizers rely on traversing the optimization landscape from random initial points, this result is especially pertinent to them. Even with gradient-free approaches, Barren Plateaus can present exponential scalability challenges, as demonstrated by their numerical simulations.

2.8 Quantum Variational Eigensolver

A hybrid classical-quantum algorithm, the variational quantum eigensolver (VQE), is configured to reduce the target's value to attain the variety of solutions for Noisy Intermediate-Scale Quantum (NISQ) computers [37]. Although its main use is in quantum chemistry, it has been modified to address issues in machine learning [71], optimization [39], physics [45, 72, 73] and finance [74, 75].

The Variational Quantum Eigensolver (VQE) was first proposed in 2014 by Peruzzo et al.[37] and formalized in [76] for estimating Hamiltonian quantum systems' ground state energy. More generally, VQE can be used to optimize any objective function that is a parameterized quantum circuit. The variational principle of quantum mechanics is the basis for VQE [77, 78]. A parameterized quantum state of the form

$$|\psi(\boldsymbol{\theta})\rangle = U(\boldsymbol{\theta})|0\rangle, \quad \boldsymbol{\theta} \in \mathbb{R}^p \tag{2.25}$$

According to the principle, the energy of the ground-state of a Hermitian operator H must satisfy

$$E_{\text{ground}} \leq \langle\psi(\boldsymbol{\theta})|H|\psi(\boldsymbol{\theta})\rangle = E(\boldsymbol{\theta}) \tag{2.26}$$

assuming that $|\psi(\boldsymbol{\theta})\rangle$ is normalized [2]. The VQE aims to find the optimal parameter vector $\boldsymbol{\theta}$ that minimizes the expectation value $E(\boldsymbol{\theta})$, resulting in an approximation of the ground-state energy and eigenstate.

One of the important tools in the application of VQE is Pauli decomposition, where a linear combination of less complex operators is used to express the Hamiltonian, which are often Pauli strings:

$$H = \sum_k \gamma_k H_k. \tag{2.28}$$

In the variational quantum eigensolver (VQE), a quantum device calculates the expectation value of each Hamiltonian term H_k, and the results are then integrated

[2] The proper relationship is

$$E_{\text{ground}} \leq \frac{\langle\psi(\boldsymbol{\theta})|H|\psi(\boldsymbol{\theta})\rangle}{\langle\psi(\boldsymbol{\theta})|\psi(\boldsymbol{\theta})\rangle}. \tag{2.27}$$

We always make sure to normalize the states $|\psi(\boldsymbol{\theta})\rangle$ in our work.

conventionally to get an estimate of the total energy of the system. To reduce this energy, a classical optimizer changes the parameters $\boldsymbol{\theta}$ in the variational ansatz over and over again.

The VQE gives both an estimate of the ground-state energy and the related eigenstate once it reaches an estimated minimum. The ground-state energy is provided by

$$E_{\text{ground}} = \min_{\boldsymbol{\theta}} \sum_k \gamma_k \langle 0|U^\dagger(\boldsymbol{\theta}) H_k U(\boldsymbol{\theta})|0\rangle \equiv \min_{\boldsymbol{\theta}} \sum_k \gamma_k E_k. \tag{2.29}$$

The term E_k is the expected value of a Pauli string measured on the quantum device for the state defined by $\boldsymbol{\theta}$. The summing over Hamiltonian terms and the optimization process take place on a classical computer.

2.9 Applications of Variational Quantum Algorithm

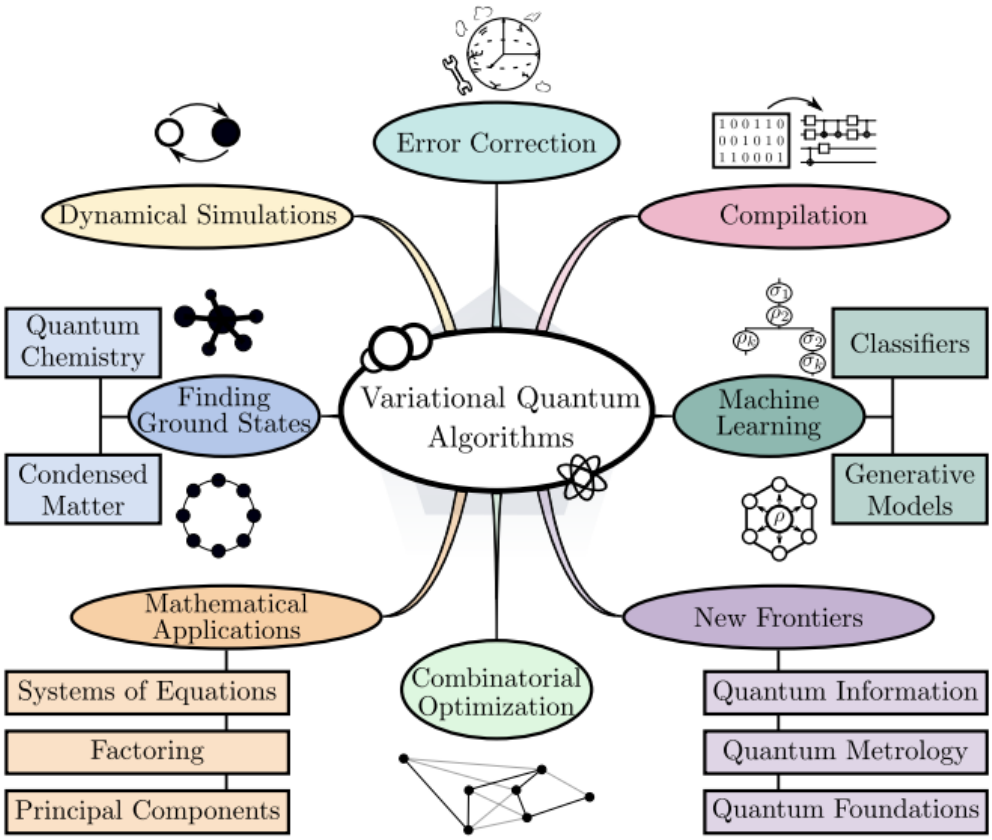


Figure 2.3: Applications of variational quantum algorithms (VQAs). Figure from Ref. [59].

Due to their applicability for NISQ devices, a large number of algorithms based on Variational Quantum Algorithms (VQAs) have been developed despite the existing

restrictions in quantum hardware. VQAs are thought to be a first step toward fault-tolerant quantum computing and show promise for a variety of activities.

Important VQA applications in fields like quantum chemistry, particle physics, and quantum machine learning. Since in 2014, Georgescu et al.'s [79]review, VQA research has expanded into a variety of areas, including quantum factoring [80], linear algebra [81] and even non-linear problems [82], which are not explicitly discussed here. Throughout this part, we intend to keep a specific focus on VQAs. However, we will also discuss how VQAs are used as sub-routines in larger algorithms. For instance, VQAs are used in many-body physics [83] to prepare a quantum system's ground state. The particular problem is then resolved using additional quantum computing techniques.

2.9.1 Quantum Chemistry

Quantum chemistry, a fast-developing topic within the interface of computational chemistry and quantum computation, tackles the constraints of traditional approaches for simulating complicated molecular systems [84, 45, 85, 86]. One major obstacle is the electronic structure problem, which aims to determine electron distribution and interactions in molecules and materials. Classical simulations become unfeasible as system size increases due to their exponential scaling. Using quantum devices, quantum chemistry approximates solutions for systems with strong correlations and entanglement that are beyond the realm of classical tractability, such as biological reactions, transition metal catalysis, and high-temperature superconductivity.

In quantum chemistry, problems are typically classified as either static or dynamic. The goal of static problems is to solve the time-independent Schrodinger equation to determine the energy spectrum, especially the ground state energy. A weighted sum of simpler Hermitian operators, typically Pauli strings, can be used to express a molecule's Hamiltonian H, as described in Eq. (2.28).

A quantum technique called Variational Quantum Eigensolver (VQE) is used to approximate the ground state energy $\tilde{E}_{\text{ground}}$. Utilizing an ansatz $|\psi(\boldsymbol{\theta})\rangle$ and minimizing the expectation value $\langle\psi(\boldsymbol{\theta})|H|\psi(\boldsymbol{\theta})\rangle$ for $\tilde{E}_{\text{ground}} \geq E_{ground}$ [84].

With prospective applications in drug discovery, materials design, and catalysis, VQAs have proven to be an excellent tool for calculating molecular energies, optimizing geometries, and modeling complex interactions in highly correlated systems.

On the other hand, dynamic simulations need to solve the time-dependent Schrodinger

equation:

$$i\frac{\partial}{\partial t}\psi(\mathbf{r},t) = H\psi(\mathbf{r},t). \tag{2.30}$$

In this case, $\psi(\boldsymbol{r},t)$ is the wave function of the system. Many methods for quantum dynamical simulations of molecular systems use the Trotter-Suzuki decomposition [87]. The Trotter-Suzuki decomposition enables quantum simulation of time development through small increments δt. This is useful for modeling transition probabilities, vibrational modes, and system responses in a variety of scenarios. In practice, however, both static and dynamical problems are frequently solved using additional approximations. For instance, the popular Born-Oppenheimer approximation is utilized in static computations.

Furthermore, accurate dynamical models of molecular systems typically require knowledge of the entire energy spectrum. If the Hamiltonian H does not depend on time, the solution to Eq. (2.30) is:

$$\psi(\mathbf{r},t) = e^{-iHt}\,\psi(\mathbf{r},t=0). \tag{2.31}$$

Using the time-independent Hamiltonian's eigenvectors $\psi_j(\boldsymbol{r})$, we can expand the above equation as follows:

$$\psi(\boldsymbol{r},t) = \sum_j e^{-iE_jt}\psi_j(\boldsymbol{r}) \tag{2.32}$$

This implies that the complete energy spectrum of the molecule must be known in order to write the solution of Eq.(2.30). In classical simulations, approximation methods like molecular dynamics are commonly employed.

In quantum computational chemistry, two main methods are used. One method is to describe the wave function as a superposition of the 2^n states:

$$|\psi\rangle = \frac{1}{\sqrt{2^n}}\sum_{j=0}^{2^n-1} a_j|x_j\rangle, \tag{2.33}$$

$|x_j\rangle$ represents the string states, i.e., $|x_0x_1.....x_{n-1}\rangle$, and $x_k \in \{0,1\}$.

The second-quantization framework is more frequently used in quantum chemistry, where Slater determinants over the fermionic Fock space are represented by the computational basis states $|i_0,i_1,....i_{n-1}\rangle$. To implement these on qubits, it is necessary to transfer fermionic creation and annihilation operators to qubit operators while maintaining their anti-commutation characteristics. This fermion-qubit mapping has been accomplished by a number of encoding schemes:

- **Jordan-Wigner [88]:** The most basic mapping is the Jordan-Wigner mapping, which maps each spin-orbital's occupancy directly to a qubit.

- **Parity mapping [89]:** This distributes occupation across numerous qubits while encoding parity onto a single qubit.

- **Bravyi-Kitaev [90]:** This mapping strikes a balance between the Jordan-Wigner and parity mapping methods which Takes parity information and occupation locality into account.

In order to simulate fermionic systems and compute the H_2 dissociation profile [85], programs such as Open Fermion [91] rely on these mappings.

In addition, quantum algorithms have been applied to bosonic systems [92], which is important for modeling superconductors [93], Bose-Einstein condensates [94], and super-fluids [95]. In the end, quantum chemistry particularly with VQAs has the potential to transform quantum system modeling and propel scientific advancement in fields where traditional methods fall short.

2.9.2 Particle Physics

Variational Quantum Algorithms (VQAs) are a promising technique in particle physics, similar to their use in computational quantum chemistry. While current quantum technology is currently limited by the Noisy Intermediate-Scale Quantum (NISQ) era [96], there is increasing hope that quantum devices in the future may be able to solve difficult particle physics issues more effectively than conventional ones. However, quantum simulations of high-energy and nuclear physics are still in their early phases, with major technical obstacles [3, 97].

VQAs could have a significant impact on comprehending processes that are hard or impossible for traditional computers to reproduce, such as nuclear matter at extreme densities in neutron stars or the early cosmos [59].

We list some of the main areas being investigated below:

Lattice Gauge Theories (LGTs)

Lattice Gauge Theories (LGTs) are crucial for studying fundamental interactions. Quantum Chromo-dynamics (QCD) and Quantum Electrodynamics (QED) are examples of quantum field theories. These models should have quantum resources that scale linearly

with lattice size, which could give them an advantage over classical simulations. Nagano et al. [98] made a significant step by simulating the Schwinger model in (1+1) dimensions, which is based on $U(1)$ theory. They simulated dynamics under electric fields and found the ground state with the Variational Quantum Eigensolver (VQE). These early-stage models have already been evaluated using ion-trap quantum devices [99, 73].

The foundation for simulating quantum field theory (QFT) on universal quantum computers was laid by Jordan et al. [100] in 2012, who showed an exponential quantum speed-up in simulating scattering processes in scalar field theory ($\lambda\phi^4$). However, error correction remains a barrier. This direction is expanded upon by the Jordan-Lee-Preskill paradigm [101]. In (1+1) dimensions, preliminary quantum simulations of gauge theories demonstrate good agreement with classical results and some Standard Model properties [102]. However, it is still theoretically difficult to simulate full (3+1) dimensional QCD, this could bring new insights into physics beyond the Standard Model. Using VQE to build a non-Abelian gauge theory on IBM-Q Casablanca to calculate hadron masses was a recent milestone by Atas et al. [103].

Nuclear Physics

VQAs have been employed in low-energy physics to calculate the ground state energies of α particles like He_4 [104] and simple nuclei like H_2, Li-H, and BeH_2 [105, 45].

In continuum quantum field theory, these problems are difficult because they require mapping infinite degrees of freedom onto a finite qubit space while maintaining gauge symmetries. There has been progress across multiple platforms. For superconducting circuits, see [106, 107], for trapped ions, see [108, 109, 73, 99]. Additional uses include running coupling constants [110], Green's functions [111], and mass gap calculations [112]. The idea that near-term quantum devices might exhibit quantum advantage in nuclear simulations is supported by these developments.

High-Energy Physics

Quantum algorithms and machine learning are being investigated in high energy physics for tasks including jet clustering [113, 114, 115], parton densities [116], parton showers [117, 118], Monte Carlo approaches [119, 120, 121], and scattering amplitudes [122, 123].

Quantum Machine Learning (QML) was recently applied to challenges such as signal-background discrimination [124, 125, 126, 127], anomaly detection [128], and

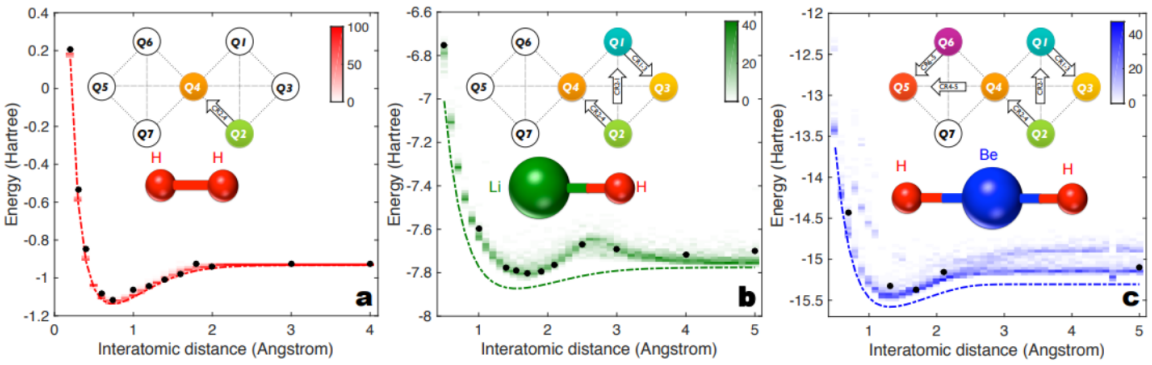


Figure 2.4: Possible energy surfaces for the molecules H_2, BeH_2, and LiH as a function of interatomic distance. The black dots represent the outcomes of digital quantum simulations on a 7 qubit system using a variational algorithm, while the dotted lines represent the precise values. A density plot derived from (classical) stochastic simulations to forecast the experimental outcomes for the used quantum algorithms is shown by the shaded area in each plot. Figure from Ref. [45].

particle track reconstruction [102, 129]. A crucial issue in collider research is determining the charge of b-jets, which Gianelle et al. [130] introduced using a quantum method. Machine learning techniques like DNNs [131], CNNs [132], RNNs [133], TNs [134], and GNNs [135] are typically used to rebuild jets, which are made up of several particles. These techniques perform better than more conventional methods like Secondary Vertex Reconstruction [136] or Track Reconstruction [137].

2.9.3 Quantum Approximate Optimization Algorithm (QAOA)

Quantum Approximate Optimization algorithm (QAOA) is a hybrid quantum-classical algorithm that was initially proposed by Farhi, Goldstone, and Gutmann in the year 2014 [39]. It is most appropriate to be utilized with near-term noisy mid-scale quantum (NISQ) hardware. These optimization problems, such as Max-Cut or Max-SAT, are usually difficult to solve classically due to their NP-hardness, making them attractive candidates for quantum speedups.

In a variational regime, the quantum approximate optimization algorithm makes use of a parameterized quantum circuit composed of two alternating unitaries: a unitary that is taken from the cost Hamiltonian H_C, where it contains the objective function to be minimized, and a unitary taken from a mixer Hamiltonian H_M, where it causes transitions between bitstring and enables exploration of the state. A lot of combinatorial optimization

problems, like Max-Cut, graph coloring, or various constraint satisfaction tasks, can be formulated as the optimization of a cost function $C(z)$ (maximize or minimize), where $z \in [0,1]^n$ is a bitstring representing a possible solution. The method sets the system up in a uniform quantum superposition of every bit string that could possibly exist, as determined by the quantum state:

$$|\psi_0\rangle = \frac{1}{\sqrt{2^n}} \sum_{z \in \{0,1\}^n} |z\rangle \tag{2.34}$$

The evolution of this state is then guided by alternating applications of the parameterized cost and mixer unitaries, expressed as

$$|\psi_p(\boldsymbol{\gamma}, \boldsymbol{\beta})\rangle = \prod_{k=1}^{p} e^{-i\boldsymbol{\beta}_k H_M} e^{-i\boldsymbol{\gamma}_k H_C} |\psi_0\rangle \tag{2.35}$$

where $\boldsymbol{\gamma} = (\boldsymbol{\gamma}_1, ..., \boldsymbol{\gamma}_p)$ and $\boldsymbol{\beta} = (\boldsymbol{\beta}_1, ..., \boldsymbol{\beta}_p)$ are real variational parameters to be optimized [138]. Subsequent to this development, the expectation value of the cost Hamiltonian is estimated by repeatedly measuring the quantum state, referred to as

$$\langle C \rangle = \langle \psi_p(\boldsymbol{\gamma}, \boldsymbol{\beta}) | H_C | \psi_p(\boldsymbol{\gamma}, \boldsymbol{\beta}) \rangle \tag{2.36}$$

A closed quantum-classical feedback loop is then created by iteratively updating $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ using a classical optimization method like COBYLA or Nelder-Mead to either maximize or minimize the $\langle C \rangle$ depending upon the problem [76, 138].

QAOA has been useful in a variety of applications, including graph Max-Cut problems, constraint satisfaction problems (CSPs), traffic flow optimization, and financial portfolio selection [36]. The approach is especially appealing to today's quantum hardware due to the fact that it employs shallow quantum circuits, which are more stable against decoherence and noise and are thus optimal for use on NISQ-era quantum processors [139].

QAOA's performance is regulated by the circuit depth parameter p, which regulates the rate of alternating uses of the operator. At $p = 1$, the performance is comparable to simple classical algorithms, as p increases, QAOA becomes capable of approaching the optimal solution more accurately, although by increasing circuit depth and classical optimization complexity. Despite its potential, QAOA has several limitations. It is highly sensitive to hardware noise, particularly as the circuit depth and qubit number rise.

Furthermore, classical parameter optimization suffers from barren plateaus, which are portions of the parameter landscape with vanishing gradients that impede convergence.

Classical optimization is a computational bottleneck in large problem cases because the parameter search gets more challenging.

Researchers are actively looking into ways to improve QAOA, such as applying machine learning models to predict good initial parameters, employing adaptive circuit depth methods, and combining QAOA with quantum neural networks to create hybrid quantum learning models. To better understand the limits of QAOA's quantum advantage, benchmarks against classical techniques such as simulated annealing and tensor network methods are performed.

2.9.4 Quantum Machine Learning

The domain of quantum machine learning (QML) links quantum computing (QC) and classical machine learning (ML), opening up new possibilities for addressing more complicated computational challenges. Traditional computing systems are approaching their physical and computational limits. This challenge is becoming more significant as global data volumes continue to grow exponentially, increasing by approximately 20% per year [140]. Through the application of quantum mechanical concepts such as quantum interference, entanglement, and superposition, quantum computing enables novel paradigms of computation within this environment and is able to accomplish tasks that would not be feasible on traditional computers.

Origins and Development of QML

QML leverages machine learning and quantum processing principles to potentially achieve a quantum advantage, even on noisy NISQ devices. In contrast to purely theoretical or long-term quantum algorithms, NISQ compatible QML techniques enhance and accelerate classical learning tasks by leveraging the intrinsic quantum properties.

Inspired by the pioneering work by Peter Wittek et al.[141], there has been a significant increase of interest in re-defining the classical machine learning models in the quantum paradigm. This development has been accelerated even further by fundamental quantum algorithms such as the Harrow-Hassidim-Lloyd (HHL) algorithm [142] or Grover's search algorithm [143]. Despite the theoretical promise of exponential speedups, real-world implementation has challenges. Near-term devices currently cannot handle the high-depth circuits needed by algorithms like HHL. To address these issues, subsequent developments have favored hybrid quantum-classical models [9].

Open Challenges and Noise Resilience

Important unanswered concerns in QML include whether quantum models are significantly better than conventional techniques and in what particular fields these benefits can be achieved. The following are notable challenges:

- **Quantum noise:**

  When present in NISQ devices, it decreases performance. Although a number of mitigation techniques are available, nothing is known about how resilient Variational Quantum Algorithms (VQAs) are to noise problems.

- **Data encoding:**

  It takes a significant amount of resources to map classical data into quantum states effectively. Polynomially efficient and hardware-adaptable encoding is necessary to prevent bottlenecks [144].

As investigated with the Quantum Approximate Optimization Algorithm (QAOA), these issues are especially important in QML applications such as binary classification.

Paradigms of QML

QML mimics traditional ML paradigms:

- **Supervised Learning (SL) [71, 145, 146]**

  In supervised learning, a labeled dataset is used to train the quantum model, and each input has a known output. Accurately predicting new data requires learning a mapping from inputs to outputs.

- **Unsupervised Learning (UL) [147, 148]**

  To find hidden structures or patterns, unsupervised learning uses training on unlabeled data. Quantum algorithms can take advantage of quantum parallelism to improve dimensionality reduction and clustering tasks.

- **Reinforcement Learning (RL) [149, 150]**

  This approach relies on agents acting in a way that maximizes the sum of their rewards. Quantum reinforcement learning investigates how entanglement and quantum states might expedite policy optimization and learning.

However, QML requires two more components: the dataset (classical or quantum) and the quantum circuit model. Based on the type of data and the computational paradigm, these are separated into the following four groups [8]:

- **CC:** The CC is the pure classical case of ML, i.e. a classical dataset that is processed with a classical computer.
- **CQ:** These algorithms deal with data from a quantum origin. The goal is to solve quantum physics problems using traditional machine learning techniques.
- **QC:** This area aims to use quantum systems and procedures, for instance, classical data is handled using variational quantum algorithms.
- **QQ:** The most ambitious and underexplored paradigm, it employs quantum algorithms to handle fundamentally quantum data.

As quantum technology advances, QQ offers long-term potential, but the QC and CQ paradigms currently predominate in QML literature.

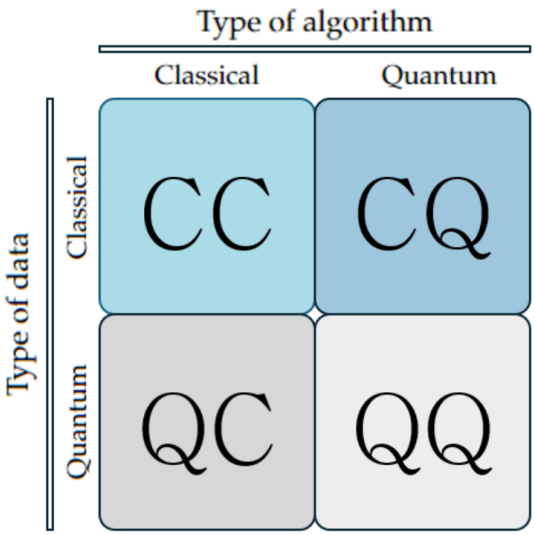


Figure 2.5: Quantum machine learning comes in four varieties. These four strategies combine machine learning and quantum computing. By differentiating between strictly classical and quantum systems of interest, the first letter denotes the type of data. For example, classical data includes examples like photographs or image databases, whereas quantum data comes from a pure quantum system. However, the second letter describes the information processing method used, distinguishing between the paradigms of Quantum for Classical (QC), Classical for Classical (CC), Classical for Quantum (CQ), and Quantum for Quantum (QQ). Figure from Ref. [8].

Data Encoding Strategies

Quantum feature mapping, or QML, is based on the encoding of traditional data into quantum states. The main strategies include:

- **Basis Encoding [146, 151]:** Basis encoding is the most natural and intuitive method that converts classical data into quantum states by mapping each classical bit to a qubit directly. An n-bit binary string is converted into an n-qubit state by the computational foundation. The traditional string 0110, for instance, is encoded as $|0110\rangle$. This technique provides a straightforward and understandable way to express quantum data.

- **Amplitude Encoding [144, 152]:** It represents information in quantum state amplitudes. Encode $x \in \mathbb{C}^n$ as follows:

$$|x\rangle = \sum_{i=0}^{2^n-1} x_i|i\rangle \quad \text{with} \quad \sum_i |x_i|^2 = 1 \tag{2.37}$$

where $|i\rangle$ is the $n-$qubit basis state, and x_i are the amplitudes associated with the state $|i\rangle$. If we want to encode the complete dataset, we must add up all of its points:

$$|\psi_S\rangle = \frac{1}{\sqrt{M}} \sum_{m=0}^{M-1} \sum_{i=0}^{2^n-1} x_i^m \, |i\rangle|m\rangle = \frac{1}{\sqrt{M}} \sum_{m=0}^{M-1} |\mathbf{x}^m\rangle|m\rangle, \tag{2.38}$$

Formally, the creation of $|\psi_S\rangle$ is comparable to embedding the $n\cdot$ A vector in an M-dimensional vector space:

$$\boldsymbol{\alpha} = x_1^1 ... x_n^1, ..., x_M^1 ... x_M^n. \tag{2.39}$$

The ability to efficiently construct the state requires both dataset encoding and single-state encoding:

$$|\psi\rangle = \sum_{i=0}^{2^n-1} \boldsymbol{\alpha}_i |i\rangle, \tag{2.40}$$

However it provides concise encoding, but its implementation is expensive.

- **Angle Encoding [153, 154, 155]:**

Each attribute x_i is represented as a rotation angle:

$$|x\rangle = \bigotimes_{i=1}^{n} \cos(x_i)|0\rangle + \sin(x_i)|1\rangle. \tag{2.41}$$

This method's 1:1 mapping between qubits and data features makes it unfeasible for huge datasets. However, it is still popular since classical data is encoded using simple single-qubit rotations that can be easily implemented.

Other approaches have been suggested, including Hamiltonian evolution ansatz encoding [156] and instantaneous quantum polynomial (IQP) style encoding [71]. However, the most effective encoding method remains unknown. For a thorough analysis of these techniques, please see [154].

Quantum Circuit Learning (QCL)

A hybrid strategy, called Quantum Circuit Learning, which is a framework for data driven ML that uses short-depth quantum circuits. This approach works for both supervised and unsupervised learning. QCL, first presented in [57], is an example of a supervised learning strategy that makes use of hybrid quantum-classical resources.

1. State preparation: Map input data $\{x_i\}$ into a quantum state $|\psi_{in}(\boldsymbol{x_i})\rangle$ to encode the information. We need to apply a unitary gate $U(\boldsymbol{x_i})$ such that $|\psi_{in}(\boldsymbol{x_i})\rangle = U(\boldsymbol{x_i})|0....0\rangle$ since quantum machines often prepare the quantum state to be $|0.....0\rangle$.

2. Parameterized transformation: Apply a parametrized unitary gate to the input state to obtain the output state, $|\psi_{out}(\boldsymbol{x_i})\rangle = U(\boldsymbol{\theta})|\psi_{in}(\boldsymbol{x_i})\rangle$.

3. Measurement: Calculate expectation values of observables, which are usually chosen as a Pauli string $\hat{P}_j \in (I, X, Y, Z)^{\otimes n}$. For F as output function, the output is $y(\boldsymbol{x_i}, \boldsymbol{\theta}) = F(\hat{P}_j(\boldsymbol{x_i}, \boldsymbol{\theta})\rangle))$

4. Optimization: To minimize the cost function $C(f(\boldsymbol{x_i}, y(\boldsymbol{x_i}, \boldsymbol{\theta}))$ of the function f and the output y_i, adjust the circuit parameters $\boldsymbol{\theta}$. Variational techniques can create variational quantum learning models (VQLMs). The cost function in these models, which in this example represents the output y, is dependent on both the parameter $\boldsymbol{\theta}$ and the input space $\boldsymbol{x}$.

$$y(\boldsymbol{x}) = \langle\psi(\boldsymbol{\theta}, \boldsymbol{x})|O|\psi(\boldsymbol{\theta}, \boldsymbol{x})\rangle, \tag{2.42}$$

5. Examine the cost function for a data set that is extracted independently from the training set to measure performance.

Variational Quantum ML Models

- **Variational Quantum Support Vector Machines (VQSVMs) [157, 153]:** These classification hyperplane problems are solved using variational techniques,

as inspired by conventional SVMs. They work with NISQ devices, in contrast to conventional QSVMs.

- **Variational Quantum Classifiers (VQCs) [158, 71]:**

  This is an important architecture for a hybrid supervised learning technique that optimizes a cost function based on labeled data and replicates the typical training/testing method.

- **Reinforcement Learning [159]:**

  To increase sample efficiency and optimize policies, quantum enhanced RL algorithms are being studied with variational architectures.

- **Transfer Learning [160]:**

  This improves generalization by enabling pretrained quantum models to apply their expertise to similar tasks.

- **Generative Models [161, 162, 163, 164]:**

  These models are used to discover the underlying data distributions. Similar to classical ML's GANs or VAEs, quantum generative models seek to sample from learnt distributions.

2.10 Conclusion

As this chapter on the applications of Variational Quantum Algorithms comes to a close, it is clear that quantum computing has the potential to revolutionize traditional computational approaches. As already stated, variational quantum algorithms, which exploit the particular features of quantum mechanics, have the potential to dramatically accelerate processing rates and capacities compared to their classical counterpart across many different areas.

In summary, variational quantum algorithms are widely applicable in various contexts, emphasizing the importance of investigating areas where they can outperform traditional techniques. However, it's important to understand that in order to fully utilize the potential of quantum computing, substantial scientific advancements and developments are required. Along with these developments, new quantum algorithms must be created and improved upon in order to fully utilize this ground-breaking technology.

Moreover, the convergence of machine learning methods with variational quantum algorithms offers intriguing prospects for speeding up challenging data analysis and optimization tasks. However, it's important to recognize that the science of quantum computing is still in its infancy and that additional research, testing, and interdisciplinary cooperation are required to completely comprehend its potential and constraints. Combining machine learning and variational quantum algorithms has the ability to transform entire sectors and address complex problems. To fully fulfill its potential, though, more investigation and study will be needed. Therefore, innovation and the realization of quantum computing's transformational potential across a range of areas depend on sustained efforts in this area.

CHAPTER 3

DIRECT MEASUREMENT AND QUANTUM NON-DEMOLITION MEASUREMENT TECHNIQUES

3.1 Scope and Organization

The variational quantum algorithm framework used in this study was described in Chapter 2, Sec. (2.1), which examined hybrid quantum-classical optimization with parameterized quantum circuits. Measurements on quantum hardware are used in this framework to obtain the information about the quantum state. Because of this, the measuring process used has a significant impact on the performance and reliability of variational algorithms.

The problem becomes more challenging because of the nature of quantum measurement. Unlike classical measurements, quantum measurements collapse the wavefunction and disturb coherent evolution [165]. Since variational algorithms repeatedly estimate expectation values and gradients with high precision, this destructive behavior creates a major practical challenge. Therefore, effective measurement procedures are crucial.

Two methods have been proposed to tackle this challenge: *Direct Measurement* (DM) and *Quantum Non-Demolition Measurement* (QNDM) [166]. To calculate analytic derivatives, we use the parameter-shift rule [167, 168], which provides an accurate and efficient approach to find the quantum-circuit gradients that are needed for training and

optimization. For each gradient component, the cost function must be evaluated at two shifted parameter values. Each evaluation requires measuring the entire observable, which is often divided into several Pauli strings. The direct-measurement (DM) method is widely considered the traditional way to estimate derivatives. It is still the standard because it is very accurate, doesn't require much extra work to set up, and works well with the noise levels of modern NISQ hardware.

Alternatively, QNDM provides a different framework inspired by weak measurements [169] and full-counting statistics [170]. In this approach, the quantum system is coupled to an ancillary detector qubit. The detector coherently stores the difference between shifted cost-function values within its phase. Measuring the detector, like a projective measurement, allows immediate access to the derivative without changing the system state. This method can reduce circuit repeats and increase resource efficiency, particularly for observables with multiple Pauli terms or higher-order derivatives [171, 172].

3.2 General Discussion

As previously stated, gradient estimation is a fundamental challenge for variational quantum algorithms that has a direct impact on their optimization efficiency and stability. To assess the efficacy of various derivative estimation methodologies, such as Direct Measurement (DM) and Quantum Non-Demolition Measurement (QNDM), it is beneficial to reference the conventional variational algorithm framework and the most common techniques for calculating cost function gradients [47, 56, 84].

First, we examine a system of n qubits in the reference state

$$|\psi_0\rangle = |0\rangle_1 \otimes |0\rangle_2 \otimes \cdots \otimes |0\rangle_n \equiv |0 \cdots 0\rangle.$$

The generic unitary transformation that acts on this state is represented by $U(\boldsymbol{\theta})$, where $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_m)$ is the vector of parameters. The quantum circuit sets up the state as

$$|\psi(\boldsymbol{\theta})\rangle = U(\boldsymbol{\theta})|\psi_0\rangle.$$

A series of parameterized gates $U_j(\theta_j)$ and non-parametrized gates V_j can be used to construct the operator $U(\boldsymbol{\theta})$, with $j \in [1, m]$, so that

$$U(\boldsymbol{\theta}) = V_m U_m(\theta_m) \cdots V_2 U_2(\theta_2) V_1 U_1(\theta_1). \tag{3.1}$$

every parametrized unitary is defined as follows: $U_j(\theta_j) = \exp(-i\theta_j H_j)$, where H_j [56]

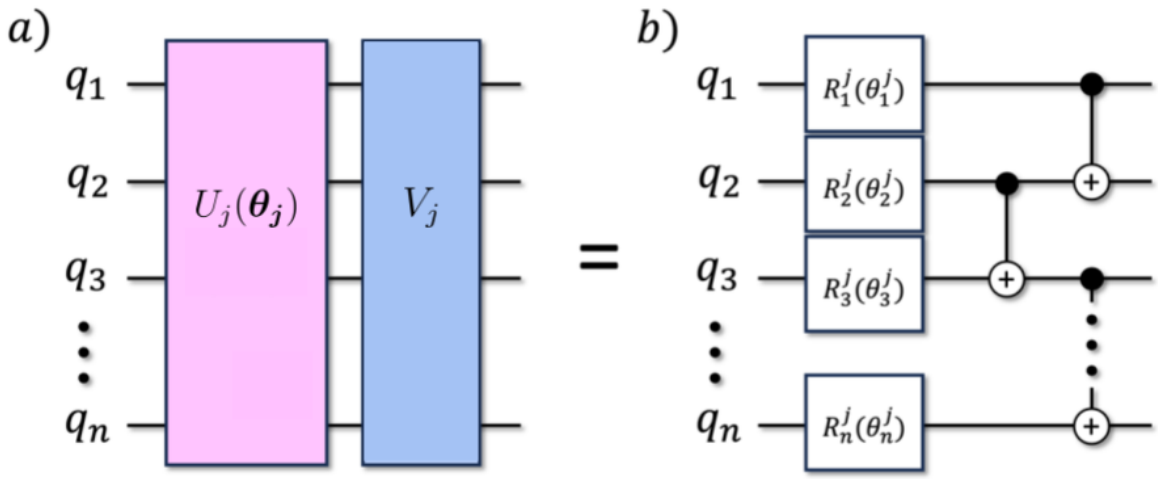


Figure 3.1: a) For a system with n qubits, the j-th layer consists of an unparameterized layer V_j and a parameterized layer $U_j(\theta_j)$. b) An illustration of a layer using two-qubit gates $C_{i-1}NOT_i$ and single-qubit gates that depend on the θ_j^i parameters. Figure from Ref. [172].

is its Hermitian generator. when $H_j^2 = 1$, this unitary can be stated as described in Eq.(2.10).

A more general scenario where the H_j do not meet the above condition is discussed in Ref.[173]. This framework covers numerous relevant scenarios, including those where H_j is a tensor product of multi-qubit Pauli operations [56].

However, to facilitate the analysis and numerical simulations, we will now limit our focus to the specific scenario depicted in Fig.(3.1).

$$U_j(\theta_j) = R_j^1(\theta_j^1) \cdots R_j^n(\theta_j^n). \tag{3.2}$$

Each parametrized single-qubit gate is represented by the operator $R_j^i(\theta_j^i)$. The independent parameter component of the transformation, denoted by V_j, can be chosen as a series of entangling gates

$$V_j = C_1NOT_2 \cdots C_{n-1}NOT_n, \tag{3.3}$$

where every $C_{i-1}NOT_i$ operator is a two-qubit gate that controls the $(i-1)$-th qubit and acts on the i-th qubit [165]. The $U_j(\theta_j)$ and V_j in order are commonly designated as a "layer" of the variational circuit.

Depending on the parameters, the operator $U(\boldsymbol{\theta})$ can be viewed in a more physical framework as being produced by a Hamiltonian $H(\boldsymbol{\theta})$. When these parameters are time

dependent, $\boldsymbol{\theta}(t) = \theta_1(t), \theta_2(t), \ldots, \theta_m(t)$, the evolution can be expressed as follows:

$$U(\boldsymbol{\theta}) = T \exp\Big\{ -i \int_0^\tau H(\boldsymbol{\theta}(t)) dt \Big\}, \tag{3.4}$$

where the time-ordering operator is denoted by T. This operator is actually a series of gates on a quantum computer.

Variational algorithms are primarily concerned with minimizing the expected value of a Hermitian observable $\hat{M}$. The cost function is given by the equation

$$f(\boldsymbol{\theta}) = \langle \psi(\boldsymbol{\theta}) | \hat{M} | \psi(\boldsymbol{\theta}) \rangle = \langle 0 | U^\dagger(\boldsymbol{\theta}) \hat{M} U(\boldsymbol{\theta}) | 0 \rangle. \tag{3.5}$$

In many applications [84, 62], a weighted sum of Pauli strings can be used to represent the observable,

$$\hat{M} = \sum_{j=1}^{J} h_j \hat{P}_j, \tag{3.6}$$

where each $\hat{P}_j$ is a tensor product of Pauli matrices acting on different qubits, and J is the number of terms. We briefly review the main tools described in Chap.2, Sec.(2.5).

A hybrid quantum-classical loop [174] is used to minimize the cost function. The process has two steps that are repeated until they converge. The first step is to use the quantum computer to calculate the cost function or its gradient at a particular point in parameter space. In the second stage, the result is sent to a classical computer. This computer employs common optimization algorithms [53, 175, 176] to figure out which way the circuit should go in parameter space to lower the cost. In this work, we concentrate on the algorithm's quantum component, which is the computation of gradients on a quantum computer.

To calculate the derivative of $f(\boldsymbol{\theta})$ with respect to θ_j, you measure the cost function at two positions that are shifted by s in the j-th direction, $\boldsymbol{\theta} + s\boldsymbol{e}_j$ and $\boldsymbol{\theta} - s\boldsymbol{e}_j$. Here, s is the shift parameter and $\boldsymbol{e}_j$ is the unit vector in the j-th direction. Then, the derivative is computed as [167, 168]

$$g_j = \frac{\partial f(\boldsymbol{\theta})}{\partial \theta_j} = \frac{f(\boldsymbol{\theta} + s\boldsymbol{e}_j) - f(\boldsymbol{\theta} - s\boldsymbol{e}_j)}{2 \sin s}. \tag{3.7}$$

This yields the parameter-shift rule [167, 168] with $s = \pi/2$. The derivation of the above parameter-shift rule expression is provided in Sec. (2.5.1).

With no approximation, this formula provides the exact derivative of $f(\boldsymbol{\theta})$. The whole gradient vector is found by doing the same thing for all directions θ_j and then sending it to the classical optimization process. It is possible to use the parameter-shift approach

to find second and higher-order derivatives of the cost function. For instance, the mixed second derivative for two parameters θ_{j_1} and θ_{j_2} can be written as four shifted evaluations (see Eq. (10) in [56]):

$$\begin{aligned} g_{j_1,j_2} &= \frac{\partial^2 f(\boldsymbol{\theta})}{\partial\theta_{j_1}\partial\theta_{j_2}} \\ &= \Big[f(\boldsymbol{\theta}+s(\boldsymbol{e}_{j_1}+\boldsymbol{e}_{j_2}))-f(\boldsymbol{\theta}+s(-\boldsymbol{e}_{j_1}+\boldsymbol{e}_{j_2})) \\ &\quad -f(\boldsymbol{\theta}+s(\boldsymbol{e}_{j_1}-\boldsymbol{e}_{j_2}))+f(\boldsymbol{\theta}-s(\boldsymbol{e}_{j_1}+\boldsymbol{e}_{j_2}))\Big]\left(2\sin^2 s\right)^{-1} \end{aligned} \tag{3.8}$$

This method can be used for derivatives of higher order [56]. The general higher order derivative looks like this:

$$g_{j_1,j_2,\ldots,j_d} = \frac{\partial^d f(\boldsymbol{\theta})}{\partial\theta_{j_1}\partial\theta_{j_2}\cdots\partial\theta_{j_d}}. \tag{3.9}$$

The computing cost of assessing higher-order derivatives increases quickly as the number of circuit evaluations increases by 2^d. You can use different classical optimization strategies to minimize the cost function with the information in these derivatives. The most frequent method is gradient descent (GD). The parameters are changed one after the other, starting with an initial set of parameters $\boldsymbol{\theta}^{(0)}$, which is sequentially updated to new values $\boldsymbol{\theta}^{(1)}, \boldsymbol{\theta}^{(2)}, \ldots, \boldsymbol{\theta}^{(T)}$. The following rule determines the parameter update:

$$\boldsymbol{\theta}^{(t)} = \boldsymbol{\theta}^{(t-1)} - \eta\nabla f(\boldsymbol{\theta}^{(t-1)}), \tag{3.10}$$

where $\eta > 0$ represents the learning rate. Newton's approach can be used if the second derivatives are given. The update rule is:

$$\boldsymbol{\theta}^{(t)} = \boldsymbol{\theta}^{(t-1)} - \eta\,[H_f(\boldsymbol{\theta}^{(t-1)})]^{-1}\nabla f(\boldsymbol{\theta}^{(t-1)}), \tag{3.11}$$

where H_f is the Hessian matrix of second derivatives. Newton's approach may be resource intensive for large parameter spaces due to the need to invert the Hessian. Second-order optimizers have been simplified to reduce the high cost of Newton's technique. The diagonal Newton optimizer, which solely uses the Hessian's diagonal entries, is one example.

Another effective approach is the quantum natural gradient optimizer [177, 56, 64]. This method utilizes the geometric information stored in the Fubini–Study metric tensor. Significantly,the metric tensor can be approximated on a quantum computer using methods akin to those employed for gradient evaluation.

3.3 Direct Measurement (DM) Technique

The easiest and most common approach for determining the value of the derivative of the cost function [178] in variational quantum algorithms (VQAs) is the Direct Measurement (DM) [55] method. The basis of this method is quite similar to classical finite-difference methods, which find the derivative of a function by looking at two locations that are close together and taking the difference ratio. In the quantum realm, the cost function for differentiation is the expectation value of a Hermitian observable $\hat{M}$ concerning a parametrized quantum state, as delineated in Eq. (2.3). Usually, a weighted sum of Pauli strings is used to express the observable $\hat{M}$,

$$\hat{M} = \sum_{j=1}^{J} h_j \hat{P}_j, \qquad \hat{P}_j = \bigotimes_{i=1}^{n} \sigma_{ij}, \qquad \sigma_{ij} \in \{I, X, Y, Z\}, \tag{3.12}$$

here J is the total number of Pauli terms that constitute the observable. Generally, we can't measure these Pauli strings at the same time since they might not commute. Instead, we have to guess each term separately. The typical way to measure is:

1. Prepare the state $|\psi(\boldsymbol{\theta})\rangle$.

2. Every qubit should undergo a basis rotation in order to match the computational basis (Z-basis) where physical measurements take place with the eigenbasis of $\hat{P}_j$. For example, to rotate from the X-basis to the Z-basis, a Hadamard gate is used.

3. Every qubit in the computational basis should be measured. One shot yields a bitstring whose parity (or other pertinent feature) yields a single sample of the eigenvalue of $\hat{P}_j$ (usually ± 1).

4. The expectation value $\langle \hat{P}_j \rangle$ can be statistically estimated by repeating steps 1-3 a large number of times (m). The final estimate of $f(\boldsymbol{\theta})$ is then produced classically as the weighted sum:

$$f(\boldsymbol{\theta}) = \sum_{j} h_j \langle \hat{P}_j \rangle.$$

After estimating the expected value, gradients can be calculated. For gates of the type $U_j(\theta_j) = e^{-i\theta_j H_j/2}$, with $H_j^2 = \mathbb{I}$, the parameter-shift rule provides an accurate formula for derivative evaluation [56], as described in Eq. (3.7). To calculate each shifted cost function $f(\boldsymbol{\theta} \pm s\boldsymbol{e}_j)$, as detailed in Eq. (3.7), the corresponding quantum circuit that performs the unitary transformation $U(\boldsymbol{\theta} \pm s\boldsymbol{e}_j)$ is executed, followed by projective measurements of

all Pauli strings found in the observable's Pauli decomposition, as specified in Eq. (3.12). These measurements are repeated for each Pauli string $\hat{P}_i$ that makes up the observable $\hat{M}$ until the required level of statistical accuracy is reached. The same steps are taken for both shifted parameter values, $\boldsymbol{\theta} + s\boldsymbol{e}_j$ and $\boldsymbol{\theta} - s\boldsymbol{e}_j$. After that, the derivative is calculated using Eq. (3.7). After this process, a single derivative g_j in the DM method can be written as

$$g_j = \sum_i \frac{h_i}{2\sin s}\Big(\mathrm{Tr}_S[\hat{P}_i U^\dagger(\boldsymbol{\theta}+s\boldsymbol{e}_j)\rho_s^0 U(\boldsymbol{\theta}+s\boldsymbol{e}_j)] - \mathrm{Tr}_S[\hat{P}_i U^\dagger(\boldsymbol{\theta}-s\boldsymbol{e}_j)\rho_s^0 U(\boldsymbol{\theta}-s\boldsymbol{e}_j)]\Big). \quad (3.13)$$

where $\rho_0 = |\psi_0\rangle\langle\psi_0|$, and Tr_S is the partial trace of all the qubits in the system. This statement makes it clear that to find one gradient component, you need to run the circuit twice and measure all of the Pauli terms each time. A quantum circuit implementation is depicted in Fig.(3.2).

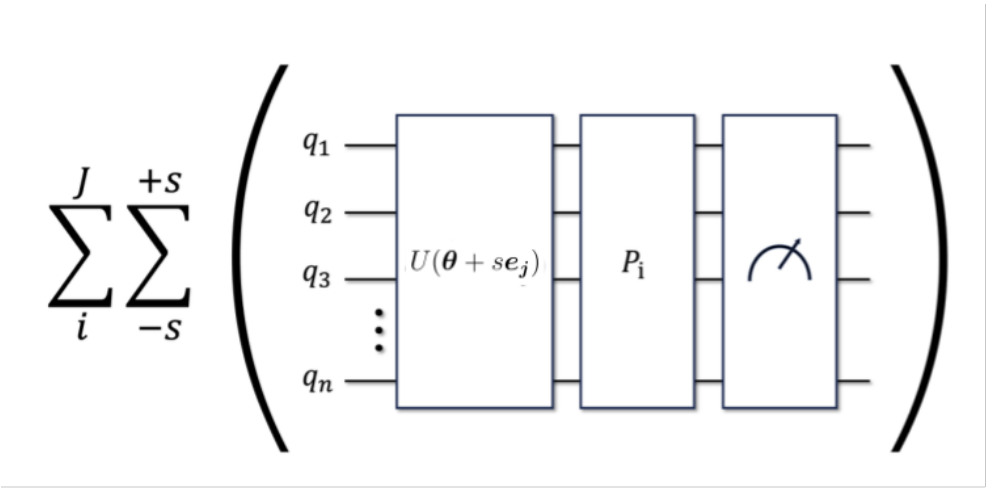


Figure 3.2: A picture of the quantum circuit that will be used to implement the DM protocol. The $U(\boldsymbol{\theta} \pm s\boldsymbol{e_j})$ transformations are the ones in Eq. (3.1). The sums show that the measure's circuits must be run for different shifts $\pm s$ and for each Pauli string $(i = 1, ..., J)$ in the observable $\hat{M}$, as shown in Eq. (3.6). Figure from Ref. [172].

The fundamental advantage of the Direct Measurement (DM) method is that it is naturally compatible with the operation of quantum computers. As DM relies on projective measurements of observables, it integrates well with optimization without requiring additional procedures. Since the most reliable and hardware-friendly function that quantum computers can carry out is measuring in the computational basis, this is particularly feasible on current NISQ technology.

3.4 Quantum Non-Demolition Measurement Approach

The Quantum Non-Demolition Measurement (QNDM) method [171] is a versatile and effective way to get derivatives of quantum cost functions by putting information in the phase of an extra detector qubit. The main concept behind this method is to connect the variational quantum system to a detector in such a way that the change in the cost function, which is related to the observable $\hat{M}$, is coherently stored in the phase of the detector. By appropriately configuring the system–detector interaction, one can obtain information regarding both $f(\boldsymbol{\theta} + s\boldsymbol{e}_j)$ and $f(\boldsymbol{\theta} - s\boldsymbol{e}_j)$ in a singular coherent process. After the interaction, we merely measure the detector qubit, and its accumulated phase immediately provides us with the cost function's derivative information.

The Quantum Non-Demolition Method uses one interferometric measurement on the detector to get the same result as the Direct Measurement (DM) method, which needs two independent circuit evaluations and projective measurements for each derivative component. This sequential coupling and single-shot measurement greatly cut down on the resources needed, which makes QNDM both theoretically interesting and practically useful [171, 172]. Consider a quantum system with the observable $\hat{M}$ and an extra qubit serving as a detector. The operator describes how the system and the detector interact [171].

$$U_{\pm} = \exp\{\pm i\lambda \hat{Z}_a \otimes \hat{M}\}, \tag{3.14}$$

Here, λ is the system–detector coupling constant, which is changed and fixed at the start of the protocol and $\hat{Z}_a$ is the Pauli-Z operator that acts on the detector qubit. The two subsystems are coupled one after the other, so that the detector first interacts with the system that has evolved under $U(\boldsymbol{\theta} - s\boldsymbol{e}_j)$ and then with the system that has developed under $U(\boldsymbol{\theta} + s\boldsymbol{e}_j)$. This simultaneous interaction lets the detector's phase gather data on how the two parameterized cost functions differ. The total unitary operator that describes how the system and detector change over time can be stated as

$$U_{\text{tot}} = e^{i\lambda \hat{Z}_a \otimes \hat{M}} U(\boldsymbol{\theta} + s\boldsymbol{e}_j) U^{\dagger}(\boldsymbol{\theta} - s\boldsymbol{e}_j) e^{-i\lambda \hat{Z}_a \otimes \hat{M}} U(\boldsymbol{\theta} - s\boldsymbol{e}_j), \tag{3.15}$$

which works on the combined Hilbert space of the system and the detector. The initial joint state is established as

$$|\Psi_0\rangle = |\psi_0\rangle \otimes \frac{|0\rangle_D + |1\rangle_D}{\sqrt{2}}, \tag{3.16}$$

where $|\psi_0\rangle$ denotes the system's initial state, and the detector qubit is initialized in an equal superposition of its computational basis states. The final entangled state $|\Psi_f\rangle = U_{\text{tot}}|\Psi_0\rangle$ encodes the change in the cost function between the two parameter configurations in the relative phase between $|0\rangle_D$ and $|1\rangle_D$ after the interaction guided by U_{tot}.

The quasi-characteristic function [171, 172, 179, 180, 181] of the detector can be used to measure the coherence of the detector states

$$\mathcal{G}_\lambda \equiv e^{i\phi(\lambda)} = \frac{{}_D\langle 0|\rho_f^D|1\rangle_D}{{}_D\langle 0|\rho_0^D|1\rangle_D}, \tag{3.17}$$

where ρ_f^D and ρ_0^D are the final and initial reduced density matrices of the detector, respectively, and $\phi(\lambda)$ is the phase accumulated by the detector throughout its interaction with the system. In terms of physics, $\mathcal{G}_\lambda$ shows how coherent the detector states $|0\rangle_D$ and $|1\rangle_D$ are. This is because it shows how the system observable $\hat{M}$ changes between the two configurations $U(\boldsymbol{\theta} \pm s\boldsymbol{e}_j)$. This function can be established experimentally by interferometric measurements, where the detector qubit is subjected to a series of a Hadamard gate (H), a phase gate (S), and another Hadamard [165], followed by projective measurement. We can directly calculate the matrix elements that make up $\mathcal{G}_\lambda$ using the

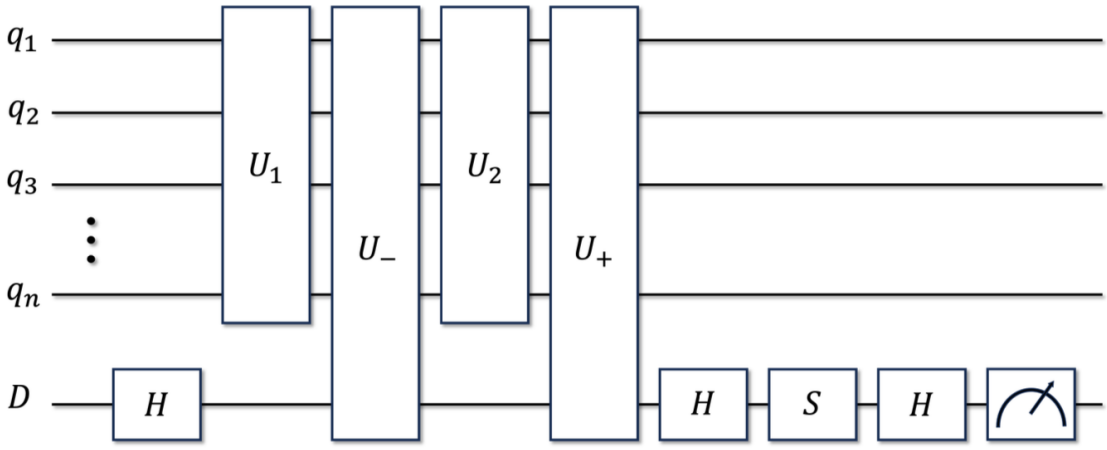


Figure 3.3: Implementation of the QNDM protocol using a quantum circuit. Here, H stands for the Hadamard gate and S for the phase gate, $U_1 = U(\boldsymbol{\theta} - s\,\boldsymbol{e}_j)$, $U_2 = U^\dagger(\boldsymbol{\theta} - s\,\boldsymbol{e}_j)\,U(\boldsymbol{\theta} + s\,\boldsymbol{e}_j)$, and $U_\pm = \exp\left\{\pm i\lambda\,\hat{Z}_a \otimes \hat{M}\right\}$ is the operator for the system–detector coupling. Figure from Ref. [172].

total evolution operator from Eq.(3.15). The final joint state's projection onto the detector basis state $|0\rangle_D$ gives us

$${}_D\langle 0|\Psi_f\rangle = \frac{1}{\sqrt{2}}\, e^{i\lambda\hat{M}} U(\boldsymbol{\theta} + s\boldsymbol{e}_j)U^\dagger(\boldsymbol{\theta} - s\boldsymbol{e}_j)e^{-i\lambda\hat{M}}U(\boldsymbol{\theta} - s\boldsymbol{e}_j)|\psi_0\rangle, \tag{3.18}$$

while projection onto the state $|1\rangle_D$ gives

$$\langle \Psi_f|1\rangle_D = \frac{1}{\sqrt{2}}\langle\psi_0|e^{-i\lambda\hat{M}}U^\dagger(\boldsymbol{\theta}+s\boldsymbol{e}_j)U(\boldsymbol{\theta}-s\boldsymbol{e}_j)e^{i\lambda\hat{M}}. \tag{3.19}$$

The top part of Eq. (3.17), which is the off-diagonal matrix element of the detector's density matrix, can be written as,

$${}_D\langle 0|\,\rho_f^D\,|1\rangle_D = \mathrm{Tr}_S[{}_D\langle 0|\,\rho_{\mathrm{tot}}\,|1\rangle_D] = \frac{1}{2}\,\mathrm{Tr}_S\left[e^{i\lambda\hat{M}}U(\boldsymbol{\theta}+s\boldsymbol{e}_j)U^\dagger(\boldsymbol{\theta}-s\boldsymbol{e}_j)e^{-i\lambda\hat{M}}U(\boldsymbol{\theta}-s\boldsymbol{e}_j)\rho_0^S\right], \tag{3.20}$$

where Tr_S represents the partial trace over the degrees of freedom of the system. Putting this result into Eq. (3.17) provides

$$\mathcal{G}_\lambda = \mathrm{Tr}_S\left[e^{i\lambda\hat{M}}U(\boldsymbol{\theta}+s\boldsymbol{e}_j)U^\dagger(\boldsymbol{\theta}-s\boldsymbol{e}_j)e^{-i\lambda\hat{M}}U(\boldsymbol{\theta}-s\boldsymbol{e}_j)\rho_0^S\right], \tag{3.21}$$

which defines the quasi-characteristic function in terms of the system observable $\hat{M}$ and the parameterized unitary operations.

$\mathcal{G}_\lambda$ is a quantum version of the classical characteristic function that is used to talk about probability distributions. In quantum physics, the corresponding distribution is a quasi-probability distribution that can assume negative or complex values [180, 179, 171].

However, its initial derivative with respect to λ remains physically meaningful and well-defined, as it corresponds to the initial moment of the observable variation. After evaluating Eq. (3.21) at $\lambda = 0$ and differentiating it with respect to λ, we get

$$\begin{aligned}-i\,\partial_\lambda\mathcal{G}_\lambda\big|_{\lambda=0} &= 2\,\mathrm{Tr}_S\left[U^\dagger(\boldsymbol{\theta}+s\boldsymbol{e}_j)\hat{M}U(\boldsymbol{\theta}+s\boldsymbol{e}_j)\rho_0^S - U^\dagger(\boldsymbol{\theta}-s\boldsymbol{e}_j)\hat{M}U(\boldsymbol{\theta}-s\boldsymbol{e}_j)\rho_0^S\right]\\ &= 2\sum_i h_i\,\mathrm{Tr}_S\left[U^\dagger(\boldsymbol{\theta}+s\boldsymbol{e}_j)\,\hat{P}_i\,U(\boldsymbol{\theta}+s\boldsymbol{e}_j)\,\rho_0^S - U^\dagger(\boldsymbol{\theta}-s\boldsymbol{e}_j)\,\hat{P}_i\,U(\boldsymbol{\theta}-s\boldsymbol{e}_j)\,\rho_0^S\right].\end{aligned} \tag{3.22}$$

This closely relates to the derivative expression that was found using the Direct Measurement approach (see Eq. (3.13)). This shows that both approaches give the same information about the gradient, but QNDM does it with just one measurement of the detector qubit.

In the small-coupling regime ($\lambda \ll 1$), the linear approximation is valid, and the quasi-characteristic function's derivative is demonstrated as

$$-i\,\partial_\lambda\mathcal{G}_\lambda\big|_{\lambda=0} = \partial_\lambda\phi(\lambda)\big|_{\lambda=0} \approx \frac{\phi(\lambda)-\phi(0)}{\lambda} = \frac{\phi(\lambda)}{\lambda} \tag{3.23}$$

where in the last steps we have used the fact that, for symmetry reason, $\phi(\theta) = 0$ (see, Ref. [171]) and $\phi(\lambda)$ is the phase of the detector, as specified in Eq. (3.17).

we can find the accumulated phase by measuring the detector populations P_0 and P_1 through an interferometric setup, where

$$\phi(\lambda) = -\arcsin(2P_0 - 1), \tag{3.24}$$

Furthermore, using Eq.(3.23) the derivative of the cost function $f(\boldsymbol{\theta})$ reads as

$$g_j \approx -\frac{\arcsin(2P_0 - 1)}{2\lambda \sin s}. \tag{3.25}$$

where the term $2\sin s$ is added to the denominator in order to directly compute the derivatives using the parameter-shift rule.

This study demonstrates that the QNDM technique retrieves identical derivative information as the DM approach, utilizing a single detector measurement instead of two whole circuit evaluations. So, QNDM gives us a clear, resource-efficient, and experimentally possible way to test gradients in variational quantum algorithms on today's noisy intermediate-scale quantum (NISQ) devices.

3.5 Practical Implementation

Fig.(3.3) illustrates a practical implementation of the proposed QNDM protocol. The circuit necessitates n logical qubits and one supplementary detector (or ancilla) qubit. The detector qubit is started in the superposition state

$$|D\rangle = \frac{|0\rangle_D + |1\rangle_D}{\sqrt{2}}$$

To distinctly differentiate the detector from the logical qubits, we will maintain the notation $\hat{Z}_a$ for the detector operator.

To compute the gradient at a certain parameter value $\boldsymbol{\theta}$, we select a small value of $\lambda \ll 1$ (see Appendix A). The comprehensive procedure to calculate the gradient of f at the position $\boldsymbol{\theta}$ is outlined as follows:

1. Choose $\lambda \ll 1$ and fix $\boldsymbol{\theta}$.
2. Implement a Hadamard gate H on the detector qubit.
3. Execute the circuit $U(\boldsymbol{\theta} - s\boldsymbol{e_j})$.
4. Couple the system and the detector by the unitary transformation $e^{-i\lambda\hat{p}\otimes\hat{M}}$.

5. Execute the circuit from $\boldsymbol{\theta} - s\boldsymbol{e_j}$ to $\boldsymbol{\theta} + s\boldsymbol{e_j}$.

6. Implement the coupling transformation $e^{i\lambda \hat{Z}_a \otimes \hat{M}}$.

7. Reapply a Hadamard gate H to the detector qubit.

8. Measure the detector qubit.

Steps (2)–(8) are repeated m times to get the detector population. The detector phase can be obtained by interferometric or tomographic measuring methods. Upon acquiring the phase, it provides the gradient of f as calculated using the QNDM technique. The computed gradient can thereafter be employed to update the parameter vector $\boldsymbol{\theta}$ in accordance with the selected optimization algorithm.

This straightforward implementation demonstrates how the QNDM technique maintains the detector qubit's separation from the logical system while enabling gradient estimation with a single quantum circuit. It illustrates the practical viability of using the QNDM protocol in variational quantum algorithms for effective parameter optimization.

3.6 Cost and Error Analysis of DM and QNDM Approaches

We have developed the theoretical basis for gradient estimation using both Direct Measurement (DM) and Quantum Non-Demolition Measurement (QNDM) methods. We now offer a detailed cost and error analysis of the two methodologies. In this section, we conduct a comprehensive cost and error analysis of the Quantum Non-Demolition Measurement (QNDM) and Direct Measurement (DM) methods. The analysis evaluates both methods performance in real world scenarios by combining theoretical resource counting with a statistical characterization of estimation errors, and physical reasoning to figure out the pros and cons of both methods in terms of how they work.

A. Cost of implementing circuits and scaling resources

A quantum algorithm's implementation cost is determined by both the number of times (shots) needed to get expectation values and the quantum resources needed to build the circuit. These include the number of logical gates, the circuit depth, the total number of qubits (including extra ones), and the difficulty of entangling processes. In today's

noisy intermediate-scale quantum (NISQ) technology, where decoherence and gate faults are still a big problem, it is important to optimize these amounts for any gradient-based optimization approach to work.

We begin by examining a general observable $\hat{M}$ that delineates the cost function in a variational method. For most quantum applications, $\hat{M}$ can be simplified as a weighted sum of Pauli operators' tensor products [84]:

$$\hat{M} = \sum_{j=1}^{J} h_j \hat{P}_j = \sum_{j=1}^{J} h_j \prod_{i=1}^{I} \sigma_i^j \tag{3.26}$$

In this case, h_j are real numbers, and σ_i^j is an operator that acts on qubit i in the j-th term of the Hamiltonian [84]. The numbers I and J are natural numbers. The qubits are numbered $i = 1, \ldots, I$ and the terms are numbered $j = 1, \ldots, J$. So, the operator $\hat{M}$ is a complex operator. Each operator $\hat{P}_j = \prod_i \sigma_i^j$ is usually referred to as a Pauli string operator. Hamiltonians like this can be used to explain many different physical systems, from quantum chemistry electronic structure models to condensed matter Hamiltonians and variational optimization landscapes [84, 43]. Usually, the length of a Pauli string is less than or equal to the number of qubits n. In most practical systems, the number of qubits that can interact with each term is also limited ($I_{\max} \leq 4$) [171, 84, 43]. This limit results from the fact that interactions in physical systems are frequently few-body in nature: Molecular Hamiltonians encompass a maximum of two-body electron interactions, whereas spin-lattice or condensed-matter models are frequently restricted to local (nearest-neighbor) couplings. So, in many circumstances that are important to physics, the scaling of J with n is polynomial, even though the complete operator space expands exponentially with the size of the system.

In both DM and QNDM techniques, the decomposition in Eq. (3.26) is essential since it dictates the method of obtaining measurement data. In the DM approach, each Pauli term $\hat{P}_j$ must be measured independently because non-commuting Pauli strings cannot be diagonalized simultaneously. This results in an overhead that increases linearly with J, since each observable needs to be estimated by running the circuit multiple times. In contrast, the QNDM protocol exploits coherent system–detector interactions to encode information about the complete operator $\hat{M}$ into a single detector-phase measurement. This significant change in how measurements are made, which significantly reduces computational cost.

The unitary operator in Eq.(3.14) describes the QNDM coupling between the system and the detector. The exponential appearing in this expression can be broken down into

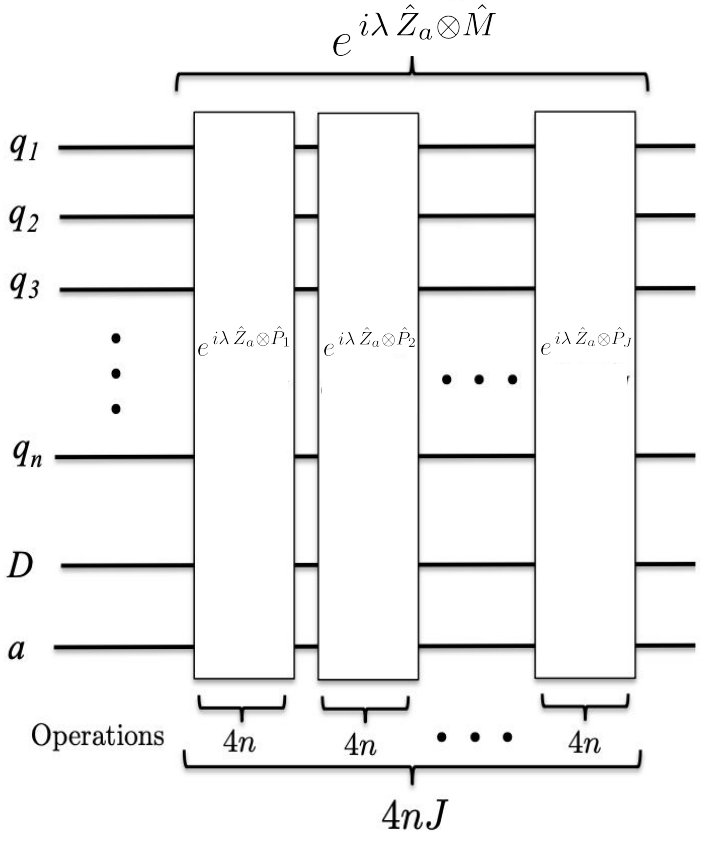


Figure 3.4: An additional ancilla qubit is used to implement the exponential operator $\exp\{i\lambda\hat{Z}_a \otimes \hat{M}\}$ on a quantum computer. (a). The operator works by breaking down into a product of Pauli string operators: $\prod_j \exp\{i\lambda\hat{Z}_a \otimes \hat{P}_j\}$. The total number of elementary operators needed is $4nJ$, where n is the number of logical qubits and J is the number of Pauli strings in $\hat{M}$. Figure from the Ref.[171].

a product of smaller unitary operators as shown in Fig.(3.4):

$$\exp\{i\lambda\hat{Z}_a \otimes \hat{M}\} = \prod_{j=1}^{J} \exp\{i\lambda h_j \hat{Z}_a \otimes \hat{P}_j\}, \tag{3.27}$$

where each factor represents a distinct Pauli string $\hat{P}_j$ with weight h_j. This decomposition has a significant computational consequence. Even though the Pauli strings may not commute, the first-order derivative information necessary for gradient estimation is maintained through this factorization, guaranteeing that QNDM yields the same linear response as the complete exponential [171].

It is possible to implement each term $\exp\{i\lambda\hat{Z}_a \otimes \hat{P}_j\}$ on a quantum processor utilizing at most $4n$ basic logical operations [165]. So, to fully connect the system and detector, $\exp\{i\lambda\hat{Z}_a \otimes \hat{M}\}$ needs about $4nJ$ gates. The QNDM circuit has two of these couplings; therefore the total number of gates for the interactions between the system and the

detector is roughly $8nJ$. Every circuit run requires a total number of$(k + 8nJ)$ gates, where k is the number of gates needed to create the parameterized ansatz $U(\boldsymbol{\theta})$. If the circuit needs to be run m times to get the right level of precision, the total cost is given by the equation

$$C_{\text{QNDM}} = (k + 8nJ)m. \tag{3.28}$$

This scaling is precise and does not depend on the small-parameter approximations frequently employed in Trotter-Suzuki expansions [84]. So, the QNDM implementation is still correct for whatever value of λh_j.

The cost analysis for the DM approach works in a different way. For each derivative component, the cost function must be evaluated twice, once at $\boldsymbol{\theta}+s\boldsymbol{e}_j$ and once at $\boldsymbol{\theta}-s\boldsymbol{e}_j$, and the whole observable $\hat{M}$ must be broken down into its Pauli strings. The total cost for one gradient component is given by the equation

$$C_{\text{DM}} = 2N_{\text{DM}}J(k + n), \tag{3.29}$$

where N_{DM} is the number of times the measurement is repeated (shots). This linear reliance on J indicates that as the Hamiltonian gets more complex, the DM approach costs more and more. Conversely, the QNDM technique collects information on all Pauli terms in one measurement of the detector phase. This means that when J gets big, the effective scaling goes down like a logarithm.

B. Error in Statistics and Mean Squared Error (MSE)

Evaluating the effectiveness and dependability of various approaches requires an understanding of how statistical mistakes impact the estimation of derivatives and gradients presented above. To assess the correctness of estimated cost function derivatives, we use Mean Squared Error (MSE). It shows the average of the squared discrepancies between the estimated gradient $\hat{g}_j$ and the real gradient g_j. The MSE shows how near the estimated values are to the real ones. Additionally, it captures the combined impact of bias and variance which are given as [182]

$$\text{MSE}(\hat{g}_j) = \text{Bias}(\hat{g}_j)^2 + \text{Var}(\hat{g}_j). \tag{3.30}$$

The mean squared error (MSE) of an estimator is the sum of its bias and variance. where $\text{Bias}(\hat{g}_j) = \mathbb{E}[\hat{g}_j] - g_j$ represents the systematic deviation (bias) and $\text{Var}(\hat{g}_j) = \mathbb{E}[(\hat{g}_j - \mathbb{E}(\hat{g}_j))^2]$ measures the random statistical fluctuations arising from finite sampling.

The bias refers to a systematic mistake in the estimated result. It indicates a constant difference from the true value that persists, even when the number of measurements or shots N [56] becomes quite large. In other words, bias shows that the way the estimate is made isn't always right.

The variance, on the other hand, shows how much the predicted values change from their average. It shows how random or noisy the outcomes are. As the number of shots N [56] increases the variance decreases, which is the opposite of bias. This is because sampling more samples usually gives a more stable and reliable estimate.

A lower MSE means that the gradient is being estimated more accurately and reliably, which has a direct effect on how quickly VQAs converge. The QNDM technique encodes derivative information in the detector's cumulative phase $\phi(\lambda)$, which comes from the quasi-characteristic function $\mathcal{G}_\lambda = e^{i\phi\lambda}$ [171, 56]. When the coupling strength is minimal, we can write $\mathcal{G}_\lambda$ as a Taylor series about $\lambda = 0$. This gives us a linear approximation whose correctness depends on how big λ is.

The bias related to this approximation can be defined as

$$\text{Bias}_{\text{QNDM}}(\hat{g}_j) = -i\left(\frac{\phi(\lambda)-\phi(0)}{\lambda} - \partial_\lambda \mathcal{G}_\lambda\right) = \frac{\lambda\,\partial_\lambda^2 \mathcal{G}_\lambda}{2} + \mathcal{O}(\lambda^2) \tag{3.31}$$

$$\text{Var}_{\text{QNDM}}(\hat{g}_j) = \sigma^2_{\text{QNDM}} \tag{3.32}$$

To find the variance σ^2_{QNDM}, you need to include the statistical error $\sigma^2_{P_0}$ that comes from measuring the detector population P_0. This error denotes the uncertainty in the measurement method. When we conduct N_{QNDM} independent single qubit measurements of the detector, the total variance is expressed as $\sigma^2 = \frac{\sigma^2_{P_0 D}}{N_{\text{QNDM}}}$, where σ_D represents the single shot variance derived from a single measurement of the detector.

By applying the conventional error propagation formulas [183] to Eq. (3.25), one may find the variance σ^2_{QNDM} from the variance of the detector population. This illustrates how measurement error at the detector level affects the overall statistical uncertainty in the QNDM calculation, given as

$$\sigma^2_{\text{QNDM}} = \frac{\sigma^2_{P_0}}{4\sin^2 s\,\lambda^2\,[1-(2P_0-1)^2]} = \frac{\sigma^2_D}{4N_{\text{QNDM}}\sin^2 s\,\lambda^2\,[1-(2P_0-1)^2]}. \tag{3.33}$$

where σ^2_D is the variance of single-shot detector outcomes, N_{QNDM} is the total number of measurements made by the detector, and P_0 is the probability of the detector population. The overall MSE for QNDM is therefore given by

$$\text{MSE}_{\text{QNDM}}(\hat{g}_j) = \frac{\lambda^2}{4}\left(\frac{\partial^2 \mathcal{G}_\lambda}{\partial\lambda^2}\right)^2 + \frac{\sigma^2_D}{4N_{\text{QNDM}}\lambda^2\sin^2 s[1-(2P_0-1)^2]}. \tag{3.34}$$

Significantly,in the $\mathrm{MSE}_{\mathrm{QNDM}}$ formulation, the variance term is directly related to λ^{-2}, whereas the bias term is directly related to λ^2. This means that the Mean Squared Error (MSE) is quite sensitive to how you choose the parameter λ. To minimize the MSE, it is essential to identify an appropriate value of λ that balances these two effects. We determined that an ideal equilibrium is achieved when $\lambda = 0.1$. For this particular value, and within the normal shot range utilized in optimization evaluations ($N \in [10^3, 10^4]$) [45, 184], the variance term σ^2_{QNDM} emerges as the predominant contribution, whereas the squared bias term remains significantly smaller. The Mean Squared Error can be estimated as follows; $\mathrm{MSE}_{\mathrm{QNDM}}(\hat{g}_j) \approx \mathrm{Var}_{\mathrm{QNDM}}(\hat{g}_j) = \sigma^2_{\mathrm{QNDM}}$.

In the DM technique, the parameter-shift algorithm gives an exact value for the derivative [56], which means there is no bias; the bias for DM is

$$\mathrm{Bias}_{\mathrm{DM}}(\hat{g}_j) = 0. \tag{3.35}$$

On the other hand, the variance results from the statistical uncertainty in measuring the observable at $\boldsymbol{\theta} + s\boldsymbol{e}_j$ and $\boldsymbol{\theta} - s\boldsymbol{e}_j$, two shifted parameter locations. The resulting variance is

$$\mathrm{Var}_{\mathrm{DM}}(\hat{g}_j) = \frac{\sigma^2_{\mathrm{DM}}(\boldsymbol{\theta} + s\boldsymbol{e}_j) + \sigma^2_{\mathrm{DM}}(\boldsymbol{\theta} - s\boldsymbol{e}_j)}{4\sin^2 s}. \tag{3.36}$$

The variance, however, arises from the statistical uncertainty linked to the measurements $\sigma^2_{\mathrm{DM}}(\boldsymbol{\theta} \pm s\boldsymbol{e}_j)$. These terms denote the variations in the evaluated functions $f(\boldsymbol{\theta} \pm s\boldsymbol{e}_j)$. In accordance with Ref. [56], we assume that the parameter shift has a negligible effect on the variance of the observed observable. Consequently, for any value of s, the relationship $\sigma^2_{\mathrm{DM}}(\boldsymbol{\theta} + s\boldsymbol{e}_j) + \sigma^2_{\mathrm{DM}}(\boldsymbol{\theta} - s\boldsymbol{e}_j) \approx 2\sigma^2_{\mathrm{DM}}$ is valid. As mentioned in sec.(3.3), to calculate the derivatives, each Pauli string must be measured separately at the parameter space locations $\boldsymbol{\theta} \pm s\boldsymbol{e}_j$ (see Eq. (3.13) and Refs. [36, 56]). The statistical error for estimating $f(\boldsymbol{\theta})$ is derived using the usual error propagation formula

$$\sigma^2_{\mathrm{DM}} = \sum_{i=1}^{J} \frac{h_i^2 \sigma_s^2}{N_{\mathrm{DM}}}, \tag{3.37}$$

here h_i denotes the parameters within the matrix $\hat{M}$ as delineated in Eq. (3.6), whereas σ_s refers to the single-shot variance derived from measurements conducted on the n qubits [56]. In Eq.(3.37), we assume that σ_s is uniform across all Pauli string measurements [56, 46]. This simplification indicates that the measurement uncertainty is consistent across all observables.

According to these assumptions, we can deduce that the Mean Squared Error (MSE) in the DM method is exclusively related to statistical noise. Consequently, the Mean

Squared Error (MSE) of the DM technique is only determined by the variance of the measurements and may be expressed as:

$$\mathrm{MSE}_{\mathrm{DM}}(\hat{g}_j) = \sum_{i=1}^{J} \frac{h_i^2 \sigma_s^2}{2N_{\mathrm{DM}} \sin^2 s}. \tag{3.38}$$

Unlike QNDM, $\mathrm{MSE}_{\mathrm{DM}}$ scales linearly with the number of Pauli terms J, indicating sensitivity to Hamiltonian size and structure. An essential observation must be made in relation to the results in Eqs. (3.34) and (3.38). The mean square error $\mathrm{MSE}_{\mathrm{QNDM}}$ is independent of the quantity or weights of the Pauli strings in $\hat{M}$, in contrast to $\mathrm{MSE}_{\mathrm{DM}}$, which is dependent on them. This distinction stems from the different methods used to extract information about the cost function f and its derivatives. The QNDM protocol stores all information about $\hat{M}$ and its Pauli strings in order through the system-detector coupling.

In the DM protocol, information about each Pauli string is gathered separately by running different circuits and taking measurements. We can accurately figure out how many shots we need to get the same level of accuracy in $\mathrm{MSE}_{\mathrm{DM}}$ and $\mathrm{MSE}_{\mathrm{QNDM}}$ using Eqs.(3.34) and (3.38). We get

$$\frac{\mathrm{MSE}_{\mathrm{DM}}(\hat{g}_j)}{\mathrm{MSE}_{\mathrm{QNDM}}(\hat{g}_j)} = 1 \;\Rightarrow\; N_{\mathrm{DM}} = N_{\mathrm{QNDM}} \frac{2\sigma_s^2 \lambda^2 \left(1 - (2P_0 - 1)^2\right)}{\sigma_D^2} \sum_i h_i^2. \tag{3.39}$$

DM method needs more measurements for bigger Hamiltonians because $\sum_i h_i^2$ goes up linearly with J. On the other hand, QNDM's shot needs to stay mostly the same.

Analyzing costs and comparing DM and QNDM methodologies

We define the cost $\mathcal{C}(g_j^i)$ (where i = QNDM, DM) to compare the two techniques. For both methods in Figs. (3.2) and (3.3), this cost shows how the number of quantum gates scales while calculating a derivative in the cost function. We also present the resource ratio $\mathcal{C}\left(\frac{g_j^{DM}}{g_j^{QNDM}}\right)$. The cost of the resources depends on the following factors:

1. the total number of Pauli strings J,
2. the total number of qubits n,
3. the total number of logical operators k employed in the circuit to get a single derivative.

Both single-qubit and two-qubit gates are included in the parameter k. As in the previous section, N_{DM} and N_{QNDM} stand for the number of shots or repetitions the DM and QNDM

methods were used to measure anything. The values in Table (3.1) come from Refs. [171, 56, 36]. It's crucial to remember that DM and QNDM require different amounts of shots. As shown in sec. (3.6), the mean squared error (MSE) for DM goes up in a straight line as the number of Pauli strings J goes up. To attain an equivalent mean squared error (MSE) in derivative estimation for both methodologies, the requisite number of shots for the DM approach must satisfy $N_{\mathrm{DM}} \propto N_{\mathrm{QNDM}} J$, as stated in Eq. (3.39).

Table 3.1: Resource cost function for DM and QNDM methods of computing first-order derivatives.

Method	**Shots**	**Cost Function**
DM	N_{DM}	$C_j^{g_{\mathrm{DM}}} = 2N_{\mathrm{DM}} J(k+n)$
QNDM	N_{QNDM}	$C_j^{g_{\mathrm{QNDM}}} = N_{\mathrm{QNDM}}(3k + 8Jn)$

3.6.1 Regime 1: When $k \gg nJ$

This regime happens when the number of logical gates k needed to implement $U(\boldsymbol{\theta})$ is significantly greater than the number of qubits n times the number of Pauli strings J. Table (3.1) and Ref.[171] show that the resource ratio is given by

$$\mathcal{C}\left(\frac{g_j^{\mathrm{DM}}}{g_j^{\mathrm{QNDM}}}\right) = \frac{2J}{3}\,\frac{2\sigma_s^2\lambda^2\left(1-(2P_0-1)^2\right)}{\sigma_D^2}\sum_{i=1}^{J} h_i^2 = \mathcal{O}(J^2). \tag{3.40}$$

This quadratic scaling with J is due to two primary things:

1. the linear computational advantage of QNDM over DM [171],

2. the fact that $\mathrm{MSE}_{\mathrm{DM}}$ depends on J, wherease $\mathrm{MSE}_{\mathrm{QNDM}}$ does not.

This is especially important for simulations of quantum chemistry. For example, in moderately complex molecule simulations [43], the number of logical gates is about $k \approx 10^9 - 10^{10}$ for implementing $U(\boldsymbol{\theta})$, the number of Pauli strings is $J > 10^3$, and the number of qubits is about $n \approx 10^2 - 10^3$. In this case, $k \gg nJ$, so the QNDM method can cut the number of resources needed by about J^2. This decrease is anticipated to increase further with larger and more intricate molecular systems. This regime is also important for quantum machine learning applications. The quantity of parameterized rotations in $U(\boldsymbol{\theta})$, or the value of k, is directly related to the expressivity of a quantum circuit, or its capacity to represent a broad range of states inside the Hilbert space [50].

3.6.2 Regime 2: When $k \ll nJ$

When the operator $\hat{M}$ consists of multiple Pauli strings, the reverse regime happens. In this situation, it's harder to guess the expectation values, but the search space in the logical circuit is smaller, thus fewer logical gates are needed. The ratio of resources in this regime is provided by

$$\mathcal{C}\left(\frac{g_j^{\mathrm{DM}}}{g_j^{\mathrm{QNDM}}}\right) = \frac{k}{4n}\,\frac{2\sigma_s^2\lambda^2\left(1-(2P_0-1)^2\right)}{\sigma_D^2}\sum_i h_i^2 = \mathcal{O}(Jk). \tag{3.41}$$

The k term shows that QNDM is cheaper than DM in this case (see Table 3.2), and the J term comes from the fact that $\mathrm{MSE}_{\mathrm{DM}}$ and $\mathrm{MSE}_{\mathrm{QNDM}}$ depend on different things. This regime is especially pertinent for numerous variational quantum algorithms (VQAs), as delineated in Ref. [185].

In summary, the QNDM procedure offers a consistent advantage in terms of accuracy and resource utilization compared to DM. It avoids having to take repeated destructive measurements of each visible term by coherently encoding gradient information in the detector's phase. This leads to a big drop in both the number of shots and the depth of the circuit, while still keeping the analytical correctness of the gradient estimation. As a result, QNDM is a better way to compute derivatives in variational quantum algorithms since it uses fewer resources and is more statistically stable. This is especially true for applications in quantum chemistry, quantum-enhanced machine learning, and high-dimensional optimization.

3.7 Computational Analysis

This part shows how the Direct Measurement (DM) and Quantum Non-Demolition Measurement (QNDM) methods compare in terms of computation. The goal is to understand how well the two approaches function in real-world scenarios and to validate the theoretical predictions that were previously stated using numerical simulations. The IBM Quantum software framework includes the IBM Aer simulator [186], which is used to run the simulations . This simulator lets you accurately and quickly simulate quantum circuits and measurement statistics without the noise that comes from real devices. This analysis examines the impact of measurement errors, circuit resources, and system parameters on both methods. We are especially interested in the mean squared error (MSE), the number of quantum operations needed, and how these things change as the system gets more complicated.

A. Error Analysis

The first section of the computer study examines statistical mistakes in both the DM and QNDM approaches. The goal is to see how the accuracy of the estimated derivatives changes when the number of measurements and the structure of the observable vary. Because both methods ultimately attempt to estimate the same physical quantity, the derivative of the cost function, they should yield the same average value for g_j, but their variance and bias may differ. The QNDM method should be better because it only needs one measurement for each gradient component, while the DM method needs several assessments at different parameter values.

The derivative g_j of the cost function $f(\boldsymbol{\theta})$ is influenced by multiple factors. We changed the following settings in the simulations:

1. The arrangement of the parameterized unitary operator $U(\boldsymbol{\theta})$, such as the kind and order of logical gates; see Eq. (3.15).

2. The total number of logical operations, represented by k, that defines the quantum circuit's depth.

3. The exact parameter direction j that the derivative is taken along.

4. The Pauli operators $\hat{P}_j$ that are part of the observable $\hat{M}$ breakdown.

5. The observable's complexity is based on how many Pauli terms J it has, as pointed out in Eq. (3.6).

We utilize a statistical average method since these factors all affect the result. Every random choice of the parameters above makes one *realization*. Each analysis point is derived by averaging over $L = 100$ independent realizations in order to guarantee statistically significant results. In each case, we consider a system of $n = 10$ qubits and perform $N = 10^4$ measurement shots per realization.

The average derivative $\mu(g_j^i)$ for the DM and QNDM techniques is shown in Fig. (3.5) with red and blue markers, respectively. The horizontal axis shows how many Pauli strings J are used in the observable. The error bars show how much the estimated derivatives vary from one realization to the next. The inset of the picture shows the ratio $\mathrm{MSE}_{\mathrm{DM}}/\mathrm{MSE}_{\mathrm{QNDM}}$, which tells us directly how accurate the two approaches are compared to each other.

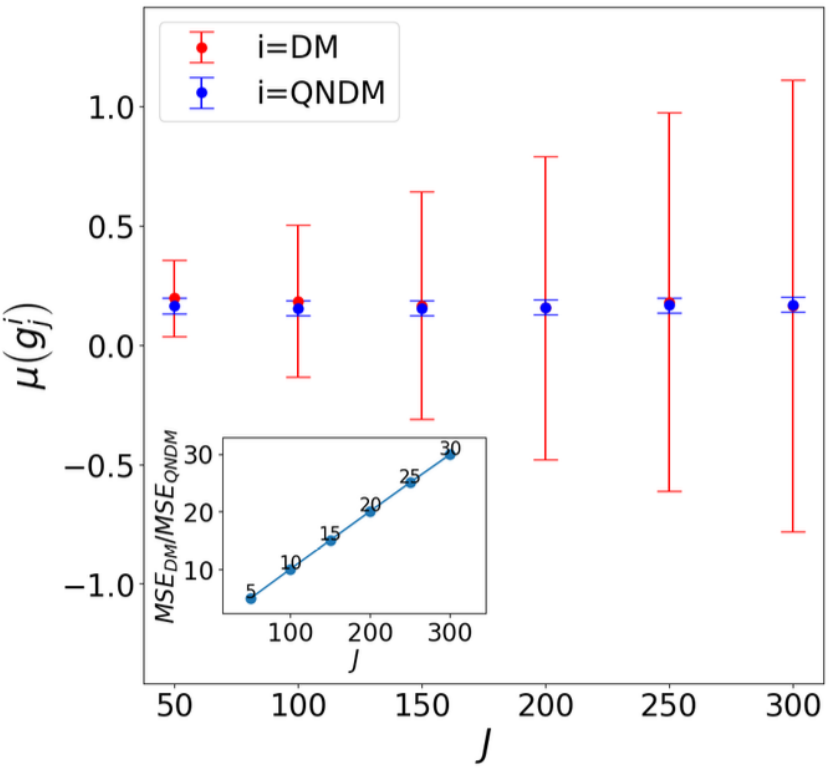


Figure 3.5: For DM (red) and QNDM (blue), the average derivative $\mu(g_j^i)$ as a function of the number of Pauli strings J. The inset shows the ratio $\mathrm{MSE_{DM}}/\mathrm{MSE_{QNDM}}$. Simulations are performed with $n = 10$ qubits, $N = 10^4$ shots, and $L = 100$ realizations. Figure from Ref. [172].

Both methods produce comparable average values for the derivative within statistical uncertainty, as demonstrated by the data in Fig.(3.5), confirming that they are objective estimators of the same physical quantity. But the two methods seem to have different ways of making mistakes. With the DM technique, the uncertainty increases as the number of Pauli strings J increases. This happens because each Pauli term $\hat{P}_j$ needs to be measured separately in the DM method, and their expectation values are only integrated at the traditional post-processing stage. As the number of independent measurements increases, so does the accumulated statistical noise, resulting in a larger variance and thus a higher mean squared error.

In comparison, the QNDM technique has a more efficient measurement process. The phase of the detector qubit encodes all information about the observable, including contributions from every Pauli term, in a coherent way through the system–detector interaction. This coherent storage stops independent measurement mistakes from building up. As a result, the variance of the QNDM estimator is essentially constant even as J

grows. You can see this benefit in the inset of Fig.(3.5), where the ratio $\mathrm{MSE}_{\mathrm{DM}}/\mathrm{MSE}_{\mathrm{QNDM}}$ goes up about linearly with J. For $J = 100$, the DM method's mean squared error is already almost ten times bigger than that of QNDM. This disparity is even more noticeable for bigger J, showing that DM needs a lot more shots to achieve the same level of accuracy. These numerical results support the theoretical prediction that the QNDM estimator has increased statistical efficiency and better scaling as the observable complexity increases.

B. Cost Analysis

Now we will examine the computing costs of the two strategies. Even though QNDM has a reduced statistical error, it's still vital to look at the actual resource needs in terms of gate operations, circuit depth, and measurement repetitions. To make a fair comparison, we set the same target precision for both approaches. This means that the MSE of each gradient component is the same. To accomplish this, the quantity of measurement shots in the DM simulations is augmented in accordance with the scaling relation established in Eq. (3.39), guaranteeing that $\mathrm{MSE}_{\mathrm{DM}} = \mathrm{MSE}_{\mathrm{QNDM}}$.

We begin by analyzing the regime $k \gg nJ$, characteristic of deep variational circuits employed in quantum chemistry. In this case, n is the number of qubits, J is the number of Pauli terms in the observable, and k is the number of logical gates needed to run $U(\boldsymbol{\theta})$. Fig.(3.6) Panel a), displays the cumulative resource cost $\mathcal{C}(g_j^i)$ for both DM and QNDM as a function of k. Every data point is an average of several random realizations that change the values of the parameters $\{J, k, \boldsymbol{\theta}, \hat{P}_j\}$. The unitary $U(\boldsymbol{\theta})$ is made up of a series of layers that alternate between parameterized single-qubit rotations $R_i(\theta_i)$ and fixed entangling operations V_j, as shown in Eqs. (3.2) and (3.3).

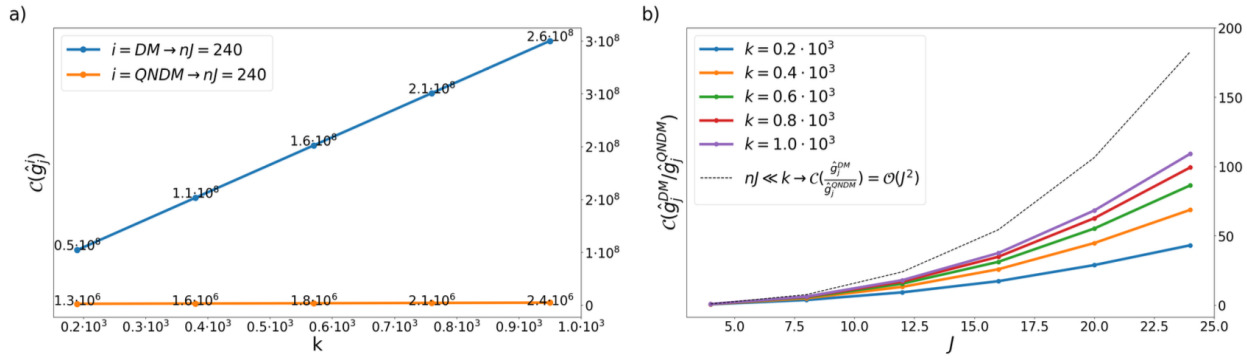


Figure 3.6: (a) Average resource cost $C(g_j^i)$ for QNDM (orange) and DM (blue) versus number of logical gates k for fixed $nJ = 240$. (b) Ratio $C_{g_j}^{\mathrm{DM}}/C_{g_j}^{\mathrm{QNDM}}$ versus number of Pauli strings J for different values of k. The simulations use $n = 10$ qubits and $L = 50$ realizations. For each case, $N_{\mathrm{QNDM}} = 500$ and N_{DM} is adjusted to match the same MSE. Figure from Ref. [172].

The findings validate that QNDM constantly necessitates fewer quantum resources than DM to attain equivalent precision. Even for circuits that aren't too deep ($k \approx 10^3$), QNDM cuts the number of logical operations by more than two orders of magnitude. Panel (b) shows how the total resource utilization ratio between DM and QNDM changes when J changes. The ratio nearly follows the theoretical scaling anticipated by Eq. (3.40), growing around as $\mathcal{O}(J^2)$. This quadratic scaling comes from two things:

1. the linear relationship between DM's MSE and J,

2. and the fact that each Pauli string needs to be assessed on its own.

The QNDM cost, on the other hand, stays virtually linear in J since the detector encodes all of the observableterms at the same time.

This benefit is considerably more important for real-world quantum chemistry simulations, where J can approach several thousand [187, 188]. For these kinds of systems, DM may need up to 10^6 times more resources than QNDM to get the same level of gradient accuracy. This shows that the QNDM method may be used with big, complicated Hamiltonians.

Next, we look at the opposite case, $k \ll nJ$, which happens in shallow circuits, quantum machine learning models, or systems where the observable is made up of a lot of Pauli terms but the quantum circuit itself is simple. Fig.(3.7) Panel a), depicts the outcomes for this scenario, demonstrating the resource cost function $\mathcal{C}(g_j^i)$ for $i = \mathrm{QNDM}, \mathrm{DM}$, employing $n = 10$ qubits and a constant $k = 0.5 \times 10^3$.

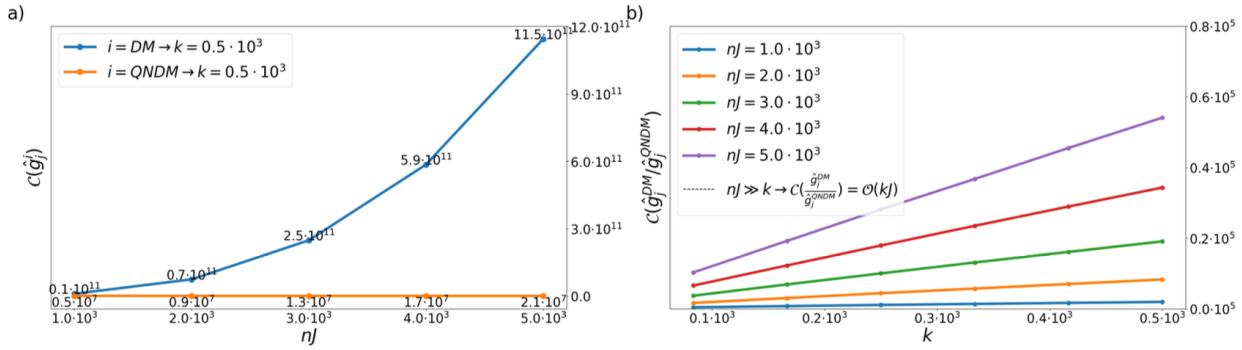


Figure 3.7: (a) Average resource cost $C(g_j^i)$ for QNDM (orange) and DM (blue) as a function of nJ for fixed $k = 0.5 \times 10^3$. (b) Ratio $C_{g_j}^{\mathrm{DM}}/C_{g_j}^{\mathrm{QNDM}}$ versus number of logical gates k for several values of nJ. Each point is averaged across $L = 50$ realizations. Figure from Ref. [172].

The same qualitative behavior is seen in this regime. The cost of the QNDM goes up in a straight line with the number of Pauli terms, while the cost of the DM goes up in a square. When $nJ \approx 10^3$, the DM approach needs around 10^4 times more resources than the QNDM method. This discrepancy gets increasingly bigger as nJ gets bigger. These data show that QNDM keeps its speed advantage no matter how deep or shallow the circuit is. It does this by coherently getting derivative information from the detector phase without having to do projective measurements again and over.

All simulations were performed on small and medium-scale systems to guarantee clarity and computational viability. In actual applications involving larger systems, such as molecular simulations or optimization problems, both the quantity of qubits and the observable complexity escalate considerably. In these realistic scenarios, the comparative advantage of QNDM is anticipated to become increasingly evident owing to its superior scalability with the quantity of Pauli strings. The computational research clearly reveals that the QNDM method is better than the old DM method in both speed and accuracy. QNDM needs fewer quantum gates, fewer measurements, and less post-processing to get the same level of accuracy. This is because the QNDM circuit keeps all of the derivative information in the phase of an extra detector qubit throughout one coherent interaction. In contrast, DM requires individual measurements for each shifted parameter and Pauli term, increasing measurement overhead and amplifying statistical noise.

In short, QNDM makes both scale and robustness a lot better. It gets more accurate results while using fewer resources, which makes it perfect for quantum processors that will be available soon. These benefits make QNDM a strong and flexible way to

estimate gradients in variational quantum algorithms and other hybrid quantum-classical optimization frameworks.

3.8 Calculation of the Second Derivative and Hessian

The evaluation of higher-order derivatives is critical in advanced variational algorithms, particularly when building second-order optimizers such as Newton or diagonal-Newton schemes. Second derivatives can be estimated using both the Direct Measurement (DM) and Quantum Non-Demolition Measurement (QNDM) approaches, however they require somewhat different amounts of resources. This part compares the two approaches in detail, using both theoretical development and cost analysis based on Refs. [171, 56, 172].

A. Expansion of the QNDM technique for second derivatives

The QNDM framework is naturally extensible to measure higher-order derivatives of the cost function. The fundamental idea is still unchanged, the phase of a detector qubit that is connected to the main system has the derivative information coherently encoded into it. For the second derivative, four sequential couplings between the detector and the system can store the information corresponding to the four points required by the second-order parameter-shift rule in the detector phase. Finally, a single-phase measurement of the detector gives the same information that would take four independent measurements in the DM method.

To do this, the total system–detector evolution operator is created as follows:

$$U_\lambda = e^{i\lambda \hat{Z}_a \otimes \hat{M}}\, U_4\, e^{-i\lambda \hat{Z}_a \otimes \hat{M}}\, U_3\, e^{i\lambda \hat{Z}_a \otimes \hat{M}}\, U_2\, e^{-i\lambda \hat{Z}_a \otimes \hat{M}}\, U_1, \tag{3.42}$$

where $\hat{M}$ is the observable of the system, and $\hat{Z}_a$ is the Pauli-Z operator acting on the ancillary detector qubit. The operators U_1, U_2, U_3, and U_4 shift the state of the system to four different places in the $(\theta_{j_1}, \theta_{j_2})$ parameter subspace [171]:

$$\begin{aligned}
U_1 &= U\big(\boldsymbol{\theta} + s(\boldsymbol{e}_{j_1} - \boldsymbol{e}_{j_2}), 0\big),\\
U_2 &= U\big(\boldsymbol{\theta} - s(\boldsymbol{e}_{j_1} + \boldsymbol{e}_{j_2}), \boldsymbol{\theta} + s(\boldsymbol{e}_{j_1} + \boldsymbol{e}_{j_2})\big),\\
U_3 &= U\big(\boldsymbol{\theta} + s(-\boldsymbol{e}_{j_1} + \boldsymbol{e}_{j_2}), \boldsymbol{\theta} - s(\boldsymbol{e}_{j_1} + \boldsymbol{e}_{j_2})\big),\\
U_4 &= U\big(\boldsymbol{\theta} + s(\boldsymbol{e}_{j_1} + \boldsymbol{e}_{j_2}), \boldsymbol{\theta} + s(-\boldsymbol{e}_{j_1} + \boldsymbol{e}_{j_2})\big).
\end{aligned} \tag{3.43}$$

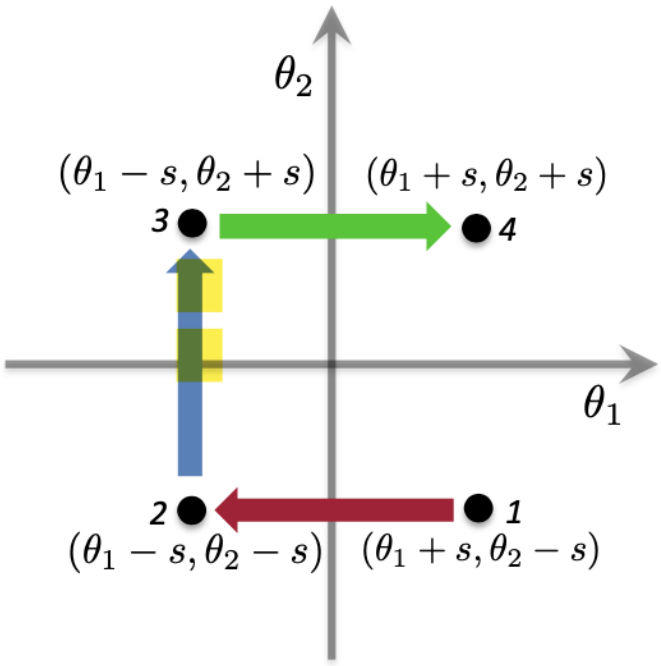


Figure 3.8: Path in the bidimensional parameter space (θ_1, θ_2) of the unitary operator $U(\boldsymbol{\theta})$ is used to compute the second derivative. We first implement $U(\boldsymbol{\theta} - s(\mathbf{e}_{j_1} + \mathbf{e}_{j_2})$, corresponding to the point $(\theta_1 - s, \theta_2 - s)$. Then, we apply a shift of $2s$ along the θ_1 coordinate to reach point 2, i.e., $(\theta_1 + s, \theta_2 + s)$. Next, we move to point 3, $(\theta_1 - s, \theta_2 + s)$, and finally to point 4, $(\theta_1 + s, \theta_2 - s)$. Note that this path is not unique: one could equivalently traverse the points in the opposite direction (1, 4, 3, 2) or start from point 3 and proceed in either orientation. Figure from Ref. [171].

There is no one way to traverse the four corners of the parameter space; changing the sequence does not change the results. When we take the derivative of the quasi-characteristic function $\mathcal{G}_\lambda$ with respect to λ at $\lambda = 0$, we obtain

$$\begin{aligned} -i\,\partial_\lambda \mathcal{G}_\lambda\big|_{\lambda=0} = f\big(\boldsymbol{\theta} + s(\boldsymbol{e}_{j_1} + \boldsymbol{e}_{j_2})\big) - f\big(\boldsymbol{\theta} + s(-\boldsymbol{e}_{j_1} + \boldsymbol{e}_{j_2})\big) \\ - f\big(\boldsymbol{\theta} + s(\boldsymbol{e}_{j_1} - \boldsymbol{e}_{j_1})\big) + f\big(\boldsymbol{\theta} - s(\boldsymbol{e}_{j_1} + \boldsymbol{e}_{j_2})\big), \end{aligned} \tag{3.44}$$

and dividing this result by $2\sin^2 s$ yields the analytical expression for the second derivative as described in Eq.(3.8). This can be access with

- a quantum circuit consisting of one detector qubit and n qubits.
- couplings between the detector and the system of the form $e^{\pm i\lambda \hat{Z}_a \otimes \hat{M}}$ that are implemented in the circuit.
- final detector phase measurement is carried out m times with an error of $1/\sqrt{m}$.

To find the second derivative using the DM method, we need to average four separate measurements, which are $M(\theta + s(\pm\mathbf{e}_{j_1} \pm \mathbf{e}_{j_2}))$. The QNDM technique, on the other hand, gets the same information with just one phase-estimation step. we can use this method to find derivatives of any order. As the order of the derivative increases, the DM approach requires more logical gates and measurements, while the QNDM approach becomes less difficult. Using the gate-counting method from Sec.(3.6), we think that calculating the gradient of f using QNDM takes at most $(k + 16nJ)m$ basic logical operations.

B. Costs Analysis of second derivatives

In the Table.(3.2), we display the costs of the resources required to calculate second derivatives. We represent the resource cost function $\mathcal{C}\left(g^{i}_{w,j}\right)$, where $i = \text{QNDM}, \text{DM}$, as the number of quantum gates required to calculate the second derivative in the w and j directions. These cost expressions are taken from [171, 56, 36]. The number of qubits n, Pauli strings J, logical operators k required to implement the unitary $U(\boldsymbol{\theta})$, and shots or circuit repeats N_i are some of the factors that affect the costs.

According to the Table.(3.2), finding a second derivative using the DM approach is twice as expensive as obtaining a first derivative. The higher cost of QNDM derives from a bigger coefficient multiplying k (which goes from 3 to 7) and from the fact that the coefficient of J is double what is needed for one derivative.

Method	Shots	Costs
DM	N_{DM}	$\mathcal{C}\left(g^{\text{DM}}_{w,j}\right) = N_{\text{DM}}\, 4J(k+n)$
QNDM	N_{QNDM}	$\mathcal{C}\left(g^{\text{QNDM}}_{w,j}\right) = N_{\text{QNDM}}\,(7k + 16Jn)$

Table 3.2: Resource cost function to calculate second derivatives using the DM and QNDM methods.

For a constant mean squared error (MSE), the ratio of the resource cost functions is articulated for two regimes as follows:

- When $k \gg nJ$, which is frequent in quantum chemistry simulations with deep circuits, the resource ratio scales as

$$\mathcal{C}\left(\frac{g^{DM}_{w,j}}{g^{QNDM}_{w,j}}\right) = \frac{4J}{7}\frac{N_{DM}}{N_{QNDM}} = \mathcal{O}(J^2) \tag{3.45}$$

Consequently, based on the quantity of Pauli strings requiring measurement, the QNDM approach may provide a substantial computational benefit. This benefit is especially important in the realm of quantum chemistry simulations.

For example, when simulating molecules of moderate complexity, we may estimate [43] that the number of operations necessary to execute the unitary $U(\boldsymbol{\theta})$ is roughly $k \approx 10^9 - 10^{10}$. The quantity of qubits typically ranges from $n \approx 10^2 - 10^3$, reflecting the expected scale of quantum computers in the near future. The Hamiltonian of such systems may comprise over $J > 10^3$ Pauli strings. Under these circumstances, we observe that $k \gg nJ$, indicating that the QNDM methodology yields a decrease of around J logical operations in comparison to the DM methodology. This renders QNDM particularly effective for extensive quantum chemistry simulations.

- In the opposite regime $k \ll nJ$,since k is usually bigger than n. The resource ratio is given by

$$\mathcal{C}\left(\frac{g_{w,j}^{\mathrm{DM}}}{g_{w,j}^{\mathrm{QNDM}}}\right) = \frac{k}{4n}\frac{N_{\mathrm{DM}}}{N_{\mathrm{QNDM}}} = \mathcal{O}(kJ). \tag{3.46}$$

In this case, the QNDM method performs better than the DM approach since the number of operations required to implement $U(\boldsymbol{\theta})$ is greater than the number of qubits n. As the derivative's order rises, this computational advantage becomes much more significant. In the DM approach, the quantity of circuit repeats increases with the order of the derivative, but in the QNDM method, the number of repetitions remains constant. Appendix B has a list of all the simulations that were used to validate these results.

In the majority of practical optimization techniques, only the first and second derivatives of the cost function are generally utilized. Nonetheless, certain research indicate that higher-order derivatives may be beneficial for tackling classically challenging optimization problems [189, 190, 171].

Moreover, in some applications, such as financial modeling, the derivatives themselves convey significant information [191]. In these instances, the QNDM method's capacity to effectively quantify high-order derivatives of the cost function offers a significant computational benefit, rendering it a formidable instrument for complex optimization and simulation tasks.

3.9 Conclusions

The technique, which is called QNDM (Quantum Non-Demolition Measurement) [166], uses a quantum detector to store relevant information in its phase without interfering with the system's evolution, and is closely related to the concept of weak values and weak measurements in quantum physics.

The primary benefit of the QNDM approach is its capacity to obtain information regarding both the gradient (first derivative) and higher order derivatives of the expectation value of a quantum observable within a single quantum circuit. This property considerably reduces the number of circuit executions required by popular optimization algorithms like gradient descent and Newton's optimization. Consequently, the total computational expense is reduced, which is especially pertinent for variational quantum algorithms (VQAs) executed on noisy intermediate-scale quantum (NISQ) devices.

From a resource standpoint, the QNDM methodology necessitates more intricate logical circuits, as the quantum detector must retain phase information during the evolution process. However, in comparison to the Direct Measurement (DM) methodology [84], the QNDM method demonstrates distinct advantages regarding resource efficiency. In particular, the QNDM approach achieves the same goal with a single circuit and a single measurement, whereas the DM approach requires two distinct quantum circuits and measurements for the gradient (and four for the Hessian).

Furthermore, in the DM technique, the statistical uncertainty (mean square error) rises linearly with the number of Pauli threads J that make up the quantum observable. In contrast, in QNDM, the uncertainty is independent of the number of Pauli strings J. Thus, for observables represented as extensive sums of Pauli strings, attaining equivalent precision with the DM technique necessitates a considerably greater number of circuit repeats.

A significant reduction in the necessary quantum resources was reported in small-scale systems and simpler quantum circuits when employing QNDM instead of DM. We anticipate that this advantage will become increasingly evident in larger and more realistic systems, such as molecular simulations and condensed-matter models [192].

The results in this chapter show that the QNDM protocol is a promising and resource-efficient alternative to classic derivative estimation methods on quantum hardware. It is especially well-suited for near-term NISQ devices because to its reduction of circuit repeats, precision, and efficiency under constrained quantum resources. These

results add to the continuous attempts to achieve quantum advantage in realistic computational problems across physics, chemistry, and optimization, and they open the door for the first real-world implementations of the QNDM algorithm.

The detailed theoretical and cost analysis presented in this chapter demonstrate that the Quantum Non-demolition measurements (QNDM) has significant benefits over the Direct Measurement (DM) technique. QNDM, in particular, shows that it is more efficient in estimating gradients and higher-order derivatives. It also has less measurement overhead and is easier to scale as the system size grows. Now, in the next chapter, we will test these results on genuine quantum systems. We will use both methods in real-world situations to test their performance, confirm the theoretical predictions, and show how strong and useful QNDM is in real quantum structures.

CHAPTER 4

IMPLEMENTATION ON MOLECULER SYSTEMS

We discussed and examined the effectiveness of the QNDM and DM approaches in the previous chapter. We analyzed the resources required and the accuracy of gradient and second derivative measurements. Based on the results, it was determined that QNDM outperforms DM in terms of efficiency and stability.

In this chapter, we moves from theoretical study to real-world application. We now use the QNDM and DM methods on real molecular systems to find their ground-state energies. The hydrogen molecule (H_2), dilithium molecule (Li_2) and the lithium hydride molecule (LiH) are chosen as test systems. These molecules are straightforward, well-characterized, and extensively utilized for evaluating quantum algorithms in electronic structure computations. Before using the QNDM and DM methods to figure out the energy of molecules, it's important to give a short summary of the molecules' physical structure and electronic properties. The analysis is supported by reliable benchmark data and discussions found in the literature [193], which also help make sure that the outcomes of this work are consistent with each other.

4.1 Molecular Systems

4.1.1 Hydrogen Molecule (H_2)

The hydrogen molecule (H_2) is the simplest, consisting of two hydrogen atoms. One electron is contributed by each atom, creating a single covalent bond. H_2 is a popular benchmark for testing the accuracy of quantum algorithms and simulation approaches due to its modest size and simple structure.

The reference study used many approaches to figure out the ground-state energy of H_2, such as the Restricted Hartree–Fock (RHF) [194], Multiplicity Hund (MH) [195], Hund subspace [196], and Full Configuration Interaction (FCI) [197] methods. The Subspace Restriction Scheme (SRS) includes the MH and Hund subspace approaches. These methods assist cut down on the number of qubits without losing accuracy. The bond length of H_2 is measured to be 0.735 Å and the energy needed to break it apart is 8.9×10^{-19} Joule. The ground-state energy calculated by both the MH and Hund subspace approaches is -4.96×10^{-18} Joule, which is the same as the value from FCI. The RHF value is a little higher, at -4.87×10^{-18} Joule. These results show that SRS-based approaches can give precise quantum results for simple molecules.

Method	Bond length (Å)	E_{diss} (J)	E_0 (J)
RHF	0.735	8.9×10^{-19}	-4.87×10^{-18}
MH Subspace	0.735	8.9×10^{-19}	-4.96×10^{-18}
Hund Subspace	0.735	8.9×10^{-19}	-4.96×10^{-18}
FCI	0.735	8.9×10^{-19}	-4.96×10^{-18}

Table 4.1: Ground-state properties of the H_2 molecule calculated using RHF (Restricted Hartree–Fock), MH (Multiplicity-based Hund subspace), Hund (Hund-rule subspace), and FCI (Full Configuration Interaction) methods. The bond length is the equilibrium distance between the two hydrogen atoms. The dissociation energy (E_{diss}) is the energy required to separate the molecule into two isolated H atoms. The ground-state energy (E_0) is the total electronic energy of the molecule at equilibrium

4.1.2 Lithium Hydride Molecule (LiH)

Lithium and hydrogen atoms combine to form the chemical known as lithium hydride LiH. It's a tiny heteronuclear diatomic molecule, which means the two atoms are distinct.

LiH possesses both ionic and covalent bonds because lithium tends to provide an electron to hydrogen. This makes it a bit more complicated than H_2.

The bond length is measured to be around 1.534 Å, while the energy of the ground state is measured to be -3.434×10^{-17} Joule using the MH [195] and Hund subspaces [197]. The FCI [197] result is -3.436×10^{-17} Joule, and the RHF energy [194] is a little higher at -3.428×10^{-17} Joule. The energy needed to break apart is about 4.360×10^{-19} Joule.

When the atoms are farther apart, there are tiny disparities between the MH and Hund subspaces, especially when they are more than 2.8 Å apart. However, both results eventually come back together near the FCI energy. This demonstrates that, even when the bond is stretched, the subspace approaches remain close to the precise quantum results.

Method	Bond (Å)	E_{diss} (J)	E_0 (J)
RHF	1.534	4.181×10^{-19}	-3.428×10^{-17}
MH Subspace	1.534	4.360×10^{-19}	-3.434×10^{-17}
Hund Subspace	1.534	4.360×10^{-19}	-3.434×10^{-17}
FCI	1.548	4.360×10^{-19}	-3.436×10^{-17}

Table 4.2: Ground-state properties of the lithium hydride (LiH) molecule calculated using RHF (Restricted Hartree–Fock), MH (Multiplicity-based Hund), Hund (Hund-rule subspace), and FCI (Full Configuration Interaction) methods. The bond length is the equilibrium distance between the Li and H atoms, E_{diss} is the dissociation energy, and E_0 is the ground-state energy.

Both H_2 and LiH are basic yet effective molecules for testing quantum simulation methods. Once we know how H_2 and LiH behave and how much energy they have, the next part uses the QNDM and DM methods on these molecules. The objective is to compute their ground-state energies utilizing both methodologies and to evaluate their efficacy regarding accuracy, resource optimization, and convergence characteristics.

4.1.3 Dilithium Molecule (Li_2)

The dilithium molecule (Li_2) is a diatomic structure made up of two lithium atoms. It is a simple but very useful model for learning about weak metallic bonding and how alkali metals act when they are in molecule form. The molecular bond is mostly made up of the overlap of these two $2s$ orbitals, which is where each lithium atom gives one valence

electron. The Li_2 bond is weaker and longer than the hydrogen bond because lithium has a bigger atomic radius and a lower ionization potential. Because of this discrepancy, the link has both covalent and metallic properties.

Li_2 is very important in quantum chemistry because it is a simple example of a metallic bond. This is a form of bond that happens when delocalized valence electrons are shared across atoms instead of firmly localized electron pairs. The molecule serves as a testing platform for quantum computational algorithms, particularly those designed to elucidate correlated electron systems and weakly bound states, where conventional mean-field approaches like Hartree–Fock sometimes inadequately address nuanced correlation phenomena.

We used a number of different computational methods to figure out its ground-state energy, such as the Restricted Hartree–Fock (RHF) [194], Multiplicity Hund (MH) [195], Hund subspace [196], and Full Configuration Interaction (FCI) [197] methods. The MH and Hund subspace methods are ways to reduce quantum space that use fewer resources while yet giving results that are as accurate as the FCI reference.

The equilibrium bond length is determined to be approximately 2.673 Å, significantly greater than that of H_2 (0.735 Å) or LiH (1.534 Å), which aligns with the weaker bonding contact. The dissociation energy of Li_2 is roughly 1.612×10^{-19} Joule, which shows that the link between the two lithium atoms is not very strong. The MH and Hund subspace approaches give a ground-state energy of about -6.535×10^{-17} Joule, which is quite close to the FCI value of -6.540×10^{-17} Joule. The RHF technique yields a higher energy of -6.527×10^{-17} Joule, which is predicted because there are no correlation effects. The computational investigation of Li_2 reveals a notable transition from straightforward

Method	Bond (Å)	E_{diss} (J)	E_0 (J)
RHF	2.673	1.612×10^{-19}	-6.527×10^{-17}
MH Subspace	2.673	1.612×10^{-19}	-6.535×10^{-17}
Hund Subspace	2.673	1.612×10^{-19}	-6.535×10^{-17}
FCI	2.673	1.612×10^{-19}	-6.540×10^{-17}

Table 4.3: Ground-state properties of the dilithium (Li_2) molecule calculated using RHF (Restricted Hartree–Fock), MH (Multiplicity-based Hund subspace), Hund (Hund-rule subspace), and FCI (Full Configuration Interaction) methods.

covalent bonding, akin to H_2, to metallic-like bonding. Li_2 is a good candidate for

testing quantum numerical differentiation algorithms since it is weakly bound, and energy precision and minor correlation effects are very important. The strong agreement between the MH, Hund, and FCI results shows that subspace restriction approaches can accurately capture delicate electron-correlation events in metallic systems without adding to the cost of computing.

Also, Li_2 is an important link between the strictly covalent bond of hydrogen and the polar covalent/ionic bond of LiH. Adding it to the molecular study helps us learn more about how QNDM and DM techniques work in many types of bonding systems, from strong covalent to weak metallic ones. The study of Li_2 not only shows that subspace-based methods work, but it also shows how quantum algorithms can handle molecules with different electronic correlation strengths.

4.2 Numerical Simulations

We investigated the performance of the **DM** and **QNDM** approaches on two kinds of problems using numerical simulations. Using a variational approach to estimate the system's *minimum (ground-state) energy* was the main goal in both cases.

We conduct all of the simulations using the Qiskit framework and its noiseless quantum simulator [198], which enables us to access algorithm performance without the impact of hardware noise. Each simulation run starts at a random point in parameter space, and optimization is done using a gradient descent method for up to 10^3 iterations, which is enough to make sure that the process converges. To account for differences caused by the starting conditions, each scenario is run ten times with various random starting positions.

We find the standard deviation and mean of the energy values from the 10 runs for each iteration. The mean energy shows how the numbers are converging, while the standard deviation gives an idea of how uncertain the numbers are. This method makes it possible to fully evaluate the DM and QNDM approaches' convergence rate, stability, and dependability.

4.2.1 Physically inspired molecular systems

Our focus is on physically inspired molecular systems, particularly H_2, Li_2 and LiH molecules. Although these systems are relatively small, it remains challenging to accurately simulate them using quantum physics. The difficulty comes from the necessity

to take into account electron correlation effects and the exact way that molecules interact with each other, which requires a lot of computing power even for molecules that look simple. Studying these systems gives us essential information about how quantum mechanics works at its most basic level and is also a good way to test and confirm new computational approaches. For example, we employed molecular symmetries to simplify the Hamiltonian, only looked at low-energy states, and included interactions between electrons in the outermost (valence) orbitals. These simplifications keep the basic physical behavior while cutting down on the expense of computing.

Following the approximations, the molecular problem was **mapped onto a quantum computer** using qubits, which are two-level quantum states [198, 199]. Each qubit stands for the quantum state of an electron or an orbital in the molecule. It is frequently hard to translate molecular Hamiltonians to qubit form because it requires many fermion-to-qubit transformations. To fix this problem quickly, we used a unique Hamiltonian database called Hamlib [199], which has qubit-based Hamiltonians that are ready to use and can be implemented directly on quantum simulators or hardware.

Hamlib: The Hamiltonian Library for Quantum Benchmarking

Hamlib (Hamiltonian Library) is a big, free dataset that was made to test quantum techniques, hardware, and software. It offers a large set of *qubit-encoded Hamiltonians* with qubit counts ranging from 2 to 1000. Hamlib was used to save researchers time, encourage results to be reproduced, and set up a common framework for standardized benchmarking in quantum computing research [199].

The library has Hamiltonians from many important scientific fields:

- **Condensed matter models**: transverse-field Ising, Heisenberg, Fermi-Hubbard, and Bose-Hubbard systems.
- **Quantum chemistry**: electronic and vibrational Hamiltonians obtained from actual molecules with precise ab-initio data.
- **Combinatorial optimization**: Some problems in combinatorial optimization are MaxCut, Max-k-SAT, Max-k-Cut, Quantum MaxCut (QMaxCut), and the Traveling Salesperson Problem (TSP).

One of the best things about Hamlib is that all of the problems are already mapped to qubits. The Hamiltonian is written as a linear combination of Pauli operations as

described in Eq.(3.12). This encoding can be used right away on quantum simulators and real quantum processors without any extra processes. To make it easy to share, Hamlib stores its Hamiltonians in the **HDF5** format and compresses them into **ZIP** archives. Each file has metadata that tells you things like:

- the total number of qubits,
- the total number of operator terms,
- and the one-norm of the operator.

This metadata is useful for determining how scaling works and what resources are required for computing. You can access the data directly with frameworks like OpenFermion or Qiskit. The most recent version (**Hamlib v1.1**, September 2024) fully supports Qiskit's SparsePauliOp representation.

The purpose of making HamLib is similar to that of standard benchmarking datasets like ImageNet for deep learning and Graph500 for supercomputing. Before Hamlib, there wasn't a big, public collection of qubit Hamiltonians. This gap is now filled by the dataset, which provides consistent benchmarks for algorithms such as **VQE**(Variational Quantum Eigensolver), **QAOA**(Quantum Approximate Optimization Algorithm), and **adiabatic quantum optimization**. Hamlib lets researchers compare algorithms and hardware platforms fairly by giving them all the same test tasks.

We used Hamlib to get the pre-encoded Hamiltonians for H_2 and LiH in this study. We were able to access the appropriate approximated Hamiltonians directly in a ready-to-use Pauli form by saying how many qubits we needed. This let us skip the manual encoding step and made sure that all simulations were the same and could be repeated.

4.2.2 Abstract Hamiltonian

The second group of problems was more "abstract and synthetic." Here, we kept the same overall Hamiltonian structure but changed the number of Pauli strings J as discussed in the Sec.(3.6) that made it up. The coefficients h_i for each phrase were picked at random from a *Gaussian distribution*. By raising J, we were able to see how the performance of algorithms and the amount of resources they need change as the system gets more complicated. These artificial Hamiltonians let us test how well DM and QNDM approaches work in a controlled fashion.

4.2.3 H_2 Molecule

We chose the hydrogen molecule (H_2) as the initial test case because it is the simplest molecule. The H_2 molecule, which has two protons and two electrons, is frequently used as a basic benchmark system in quantum chemistry to test quantum algorithms. Because of its simple structure and small size, the exact ground state energy of this system can be easily determined using both classical and quantum simulators. This makes it a good test case for novel methods like DM and QNDM. [1]

The goal of the simulation is to minimize the expectation value of the Hamiltonian by using a variational technique to find the ground-state energy. We got the Hamiltonian for this molecule from the Hamlib database [199], which has qubit-encoded Hamiltonians that are ready to use for many different compounds and models. The Hamiltonian in this example is made up of $J = 15$ Pauli terms and needs $n = 4$ qubits to be mapped to qubits. These terms account for all relevant electron interactions inside the minimal basis set after the application of physical symmetries and the stabilization of core orbitals.[2].

Simulation parameters: We used $N_{\text{shots}} = 10^3$ for both the DM and QNDM methods. This is the number of times we measured each expectation value. There were $L = 5$ variational layers in the quantum circuit, and the classical optimizer used gradient descent with a learning rate of $\eta = 0.1$. The coupling constant for QNDM is set to $\lambda = 0.1$.

Convergence behavior: Fig.(4.1) displays the effects of minimizing the energy of the H_2 molecule. The solid lines show the average convergence curves obtained with the DM (blue) and QNDM (orange) methods, and the shaded areas show the associated statistical uncertainty that comes from random starting values. we can lower this uncertainty by taking more measurement shots (which lowers sampling noise) or by making the circuit deeper with more layers, which makes the variational ansatz more expressive. After roughly 10^3 repetitions, the DM and QNDM approaches both reach the same minimum energy, and the uncertainty bands almost fully overlap. This shows that both techniques work equally well for this simple system in terms of accuracy and speed of convergence.

[1] The hydrogen molecule is the standard "hello world" system for the variational quantum eigensolver (VQE) and other similar algorithms. It has been utilized in numerous experimental demonstrations of quantum computing, including the inaugural VQE implementation by Peruzzo ([37]) *et al.*, *Nature Communications* (2014).

[2] A "basis set" in quantum chemistry is a group of atomic orbitals that are used to describe how the electrons in a molecule are arranged. A basic set for H_2, like STO-3G, employs one orbital for each hydrogen atom. This results in a 4-qubit representation after symmetry reduction

Comparing resources: Even if both strategies converge in a similar way, the amount of

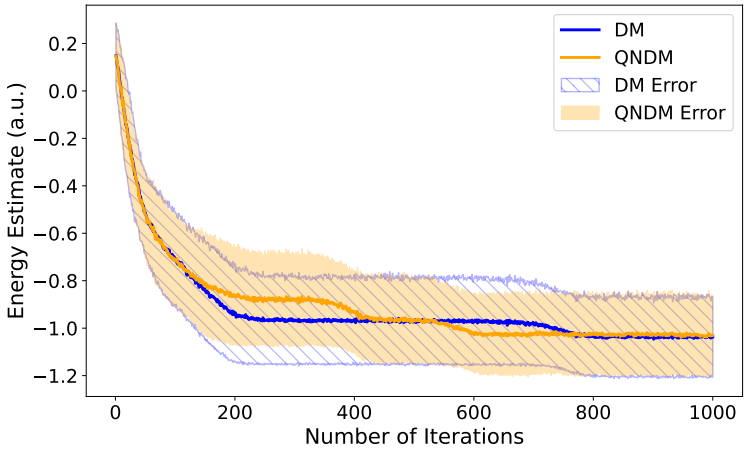


Figure 4.1: The DM method (blue) and the QNDM approach (orange) are used to minimize the energy of H_2. Solid lines represent the average convergence over multiple simulations from different initial points, while shaded regions show the corresponding statistical uncertainty. For Simulations we employed a gradient descent optimizer with learning rate $\eta = 0.1$, circuit depth $L = 5$, and $N_{\text{shots}} = 10^3$, and coupling constant for QNDM $\lambda = 0.1$.

computing power they need is very different. The total number of logical quantum gates required for the DM method in order to compute the energy gradients is

$$R_{\text{DM}} = 2.14 \times 10^7,$$

whereas for the QNDM approach it is

$$R_{\text{QNDM}} = 6.20 \times 10^6.$$

This corresponds to a reduction of approximately

$$\frac{R_{\text{DM}} - R_{\text{QNDM}}}{R_{\text{DM}}} \approx 70\%.$$

This 70% drop in the number of logical gates is a big step forward for the QNDM method's use of quantum resources.[3]

[3] A "logical quantum gate" is a theoretical function that an ideal quantum computer might be able to do without making mistakes. In practical devices, error-correcting codes protect many physical gates that make up each logical gate. So, cutting down on the number of logical gates also indirectly lowers the need for hardware, circuit depth, and noise buildup.

In short, the H_2 molecule makes it easy to compare how well DM and QNDM approaches work. Both methods get to the same lowest energy level, but QNDM uses a lot less computing power. It cuts the number of quantum gates by about 70% while keeping the same level of accuracy. This indicates that QNDM not only gives accurate findings but also works more efficiently. This makes it a very attractive method for future quantum simulations of bigger and more complicated molecules, where limited hardware resources are still a big problem.

4.2.4 LiH Molecule

We present the lithium hydride (LiH) molecule as a second example. LiH is a straightforward heteronuclear molecule; however, it is significantly more complex than H_2. It has more electrons and orbitals, which makes it a good next step for testing how well the DM and QNDM approaches work on systems that need more qubits and more Hamiltonian terms. LiH is a common benchmark molecule in quantum chemistry and has been extensively utilized to evaluate variational quantum eigensolver (VQE) algorithms on both simulators and actual quantum hardware. The LiH molecule is often employed in quantum chemistry benchmarks because its electronic structure is small but not simple; therefore, electron correlation must be captured correctly.

We got the molecular Hamiltonian for LiH from the Hamlib database [199]. This database lets users choose between multiple approximate Hamiltonians by picking the number of qubits n. Adding more qubits makes the calculations more accurate, but it also adds more Hamiltonian terms J and makes the whole thing more expensive. We chose $n = 10$ qubits for this simulation to keep a good balance between accuracy and resource use. This configuration corresponds to an approximate Hamiltonian with $J \approx 300$ Pauli-string terms, striking a favorable balance between accuracy and efficiency for near-term quantum devices.

Simulation parameters: We chose $N_{\text{shots}} = 10^3$ for both techniques. This tells us how many measurements we need to make for each expectation-value evaluation. We employed a gradient descent optimizer with a learning rate of $\eta = 0.5$. For the QNDM technique, the coupling constant was set at $\lambda = 0.4$. The number of layers in the circuit for this simulation was $L = 5$, which is the same as for the H_2 case.

Behavior of convergence: Fig. (4.2) shows the results of the energy minimization for LiH. The blue line shows the DM approach, and the orange line shows the QNDM

method. The solid lines show the average convergence of the energy across 10 runs with varied random starting values. The shaded areas illustrate the statistical uncertainty that comes with those runs. Like H_2, both algorithms find the same minimal energy and have similar levels of uncertainty. This shows that they work well and consistently over a range of beginning conditions. **Comparison of resources:** Even though both methods come

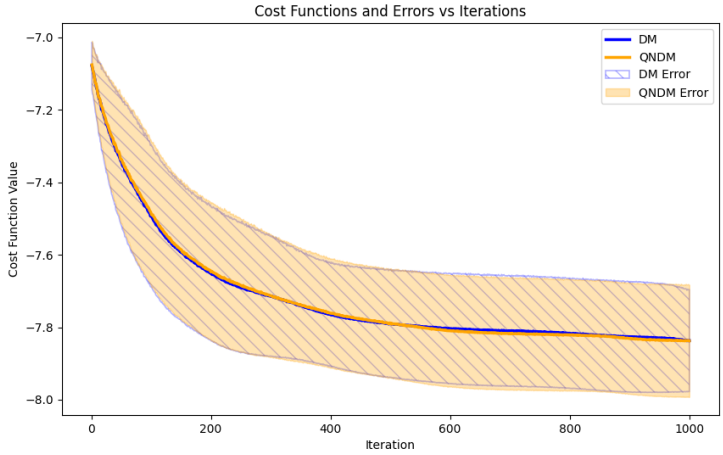


Figure 4.2: Energy minimization of LiH with the QNDM approach (orange) and the DM method (blue). Solid lines represent the average energy over multiple simulations with different initial points, while shaded areas show the statistical uncertainty. For Simulations we employed a gradient descent optimizer with learning rate $\eta = 0.5$, circuit depth $L = 5$, and $N_{\text{shots}} = 10^3$, and coupling constant for QNDM $\lambda = 0.4$.

to the same conclusion, the costs of computing them are very different.

$$C_{\text{DM}} \approx 2.62 \times 10^{12},$$

whereas for the QNDM approach it was

$$C_{\text{QNDM}} \approx 5.46 \times 10^{11}.$$

This means that the amount of computing power needed has gone down by over 79%. This kind of reduction becomes more useful for bigger molecules, as the number of qubits and Hamiltonian terms grows quickly, making the circuit depth and gate counts grow exponentially. The results demonstrate that the QNDM approach is more accurate and scales better than the DM approach.

Discussion and expansion: In quantum simulations, it's very important to find the right balance between accuracy and cost. The quality of the result and the resources

needed depend on how many qubits n and how deep the circuit L are. For LiH, $n = 10$ and $L = 5$ are good test points for quantum hardware simulators that are currently available. The outcomes for these systems were uniform: the QNDM methodology provided similar precision to DM while significantly decreasing the total count of logical quantum gates needed.

In short, the LiH molecule shows how both DM and QNDM act in a more complicated way than H_2. Both approaches got to the same minimum energy, however QNDM did it with about 79% less computing power. This shows that QNDM is not only accurate but also scalable, which makes it a good candidate for simulating bigger molecular systems where the limited resources of quantum devices in the near future are a big problem.

4.2.5 Li_2 Molecule

The Li_2 molecule (dilithium) is a diatomic system made up of two lithium atoms. It has a basic structure, but it is more complicated than H_2 and LiH. It is often used as a test case for medium-sized quantum chemistry problems. Li_2 has a bond length of roughly 2.673 Å in its ground state, and its valence electrons are weakly covalently bonded.[4] Li_2 needs to accurately capture electron–electron correlation effects since both lithium atoms give up one valence electron. This is an excellent way to test how the DM and QNDM approaches work in systems that need more qubits and Hamiltonian terms than the smaller molecules.

Parameters for the simulation: The molecular Hamiltonian for Li_2 was obtained from the **Hamlib** database [199], which offers qubit-based Hamiltonians with varying degrees of approximation. We employed $n = 10$ qubits and about $J \approx 300$ Pauli-string terms for this study. The accuracy and computational cost are well-balanced in this configuration. We employed $N_{\text{shots}} = 10^3$ and a circuit depth of $L = 5$ for both the DM and QNDM simulations. For the QNDM technique, we set the coupling constant at $\lambda = 0.99$ and used a gradient descent optimizer with a learning rate of $\eta = 0.2$. Stable convergence was achieved under these conditions without requiring an excessive amount of processing power.

Behavior of convergence: Fig.(4.3) shows the results of minimizing energy for Li_2. The orange line shows the QNDM approach, and the blue line shows the DM method. The solid lines show the average convergence of the energy levels from several simulations

[4]Information from the NIST Chemistry WebBook: https://cccbdb.nist.gov/exp2x.asp?casno=14452596

that started with different random initial parameters. The shaded areas demonstrate how much uncertainty there is in the statistics when you use different starting points. Both methods exhibit smooth and stable convergence toward the same minimum energy, although the QNDM curve gets there sooner and with less uncertainty. This illustrates that QNDM makes optimization faster and more stable without changing the level of accuracy. **Resource comparison:** The parameter-shift approach was used to find the

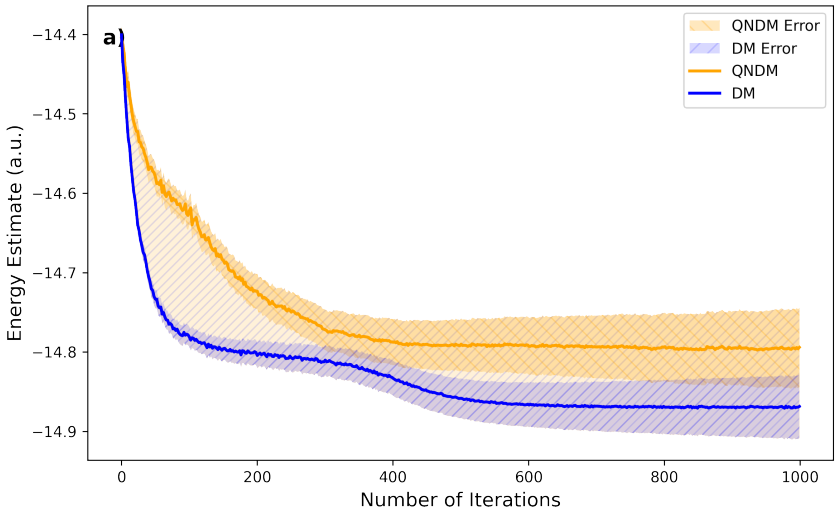


Figure 4.3: Energy minimization of Li_2 using the QNDM approach (orange) and the DM method (blue).Solid lines represent the average convergence curves over multiple simulations from different initial points, while shaded areas represent the statistical uncertainty. For Simulations we set the number of shots to $N_{\text{shots}} = 10^3$, the circuit depth to $L = 5$, the learning rate for the gradient descent optimizer to $\eta = 0.2$, and the coupling constant for QNDM to $\lambda = 0.99$.

gradients of the energy with respect to the circuit parameters at each optimization stage. For Li_2, the number of variational parameters and the circuit's depth make it hard to estimate the gradient. But QNDM cuts down on the overall amount of logical operations and gradient evaluations that are needed. The total number of logical quantum operations for the DM method was roughly

$$C_{\text{DM}} \approx 4.1 \times 10^{12},$$

while for the QNDM approach it was around

$$C_{\text{QNDM}} \approx 1.2 \times 10^{12}.$$

This corresponds to a reduction of nearly

$$\frac{C_{\text{DM}} - C_{\text{QNDM}}}{C_{\text{DM}}} \times 100\% \approx 70.7\%.$$

So, QNDM gets the same final energy with fewer logical processes and gradient evaluations.

As the size of the system grows, the number of qubits n and Pauli strings J grows quickly. This makes the parameter space bigger and the circuit deeper. With $n = 10$ and $J \approx 300$, the simulation for Li_2 is already in a difficult range. Here, the efficiency of QNDM is even more crucial. Compared to DM, it converges faster, has fewer gate operations, and has less uncertainty. This lessening of computing cost makes QNDM better for quantum hardware that is close to being built, when gate depth and noise are big problems.

In short, the Li_2 molecule shows that QNDM is still better than DM as molecular systems get bigger and more complicated. Both approaches get to the same minimal energy, but QNDM does so with about 71% fewer logical quantum gates and converges faster. This means that QNDM is a promising and scalable way to run quantum chemistry simulations on future quantum computers, where using resources wisely is very important.

4.2.6 The Large Hamiltonian

In this final part, we examine how the DM and QNDM approaches function when the Hamiltonian becomes large and complex. In the HamLib database, increasing the Hamiltonian complexity, as indicated by the number of Pauli strings J, is often associated with increasing the number of qubits n. But having more qubits quickly makes the calculations very hard. Simulations take longer and use more memory, especially if we need to track the whole convergence process and do repeated runs to get reliable statistical averages. We utilized a simplified but physically meaningful model that lets J expand while keeping the number of qubits fixed to get around these problems and still examine the scaling behavior.

Description of the system: We looked at a random Hamiltonian that was the same as the one in Eq. (3.12), but this time, each of the Pauli strings' coefficients h_i was derived from a Gaussian distribution, which had a mean of $\mu = 1$ and a standard deviation of ($\sigma = 0.1$). By increasing J without altering n, this technique gives us direct control over the complexity of the Hamiltonian. All of the simulations were conducted using $n = 10$ qubits, $N_{\text{shots}} = 10^3$, and a circuit depth of $L = 5$. We used a gradient descent optimizer with a learning rate of $\eta = 0.05$. For the QNDM technique, the coupling constant was adjusted at $\lambda = 0.01$. These numbers make the computational cost minimal while still

allowing for reliable convergence.

Convergence behavior: The energy minimization results for Hamiltonians with $J = 750$ and $J = 1000$ are displayed in Fig.(4.4). In both situations, the DM (blue) and QNDM (orange) methods reach the same lowest energy after about the same number of rounds. The shaded areas in the illustration show the statistical uncertainty that comes from running simulations with varied random starting parameters. Both approaches exhibit stable convergence and levels of uncertainty that are similar. The QNDM technique, on the other hand, shows smoother convergence and slightly smaller statistical fluctuations, which means that the numbers are more stable.

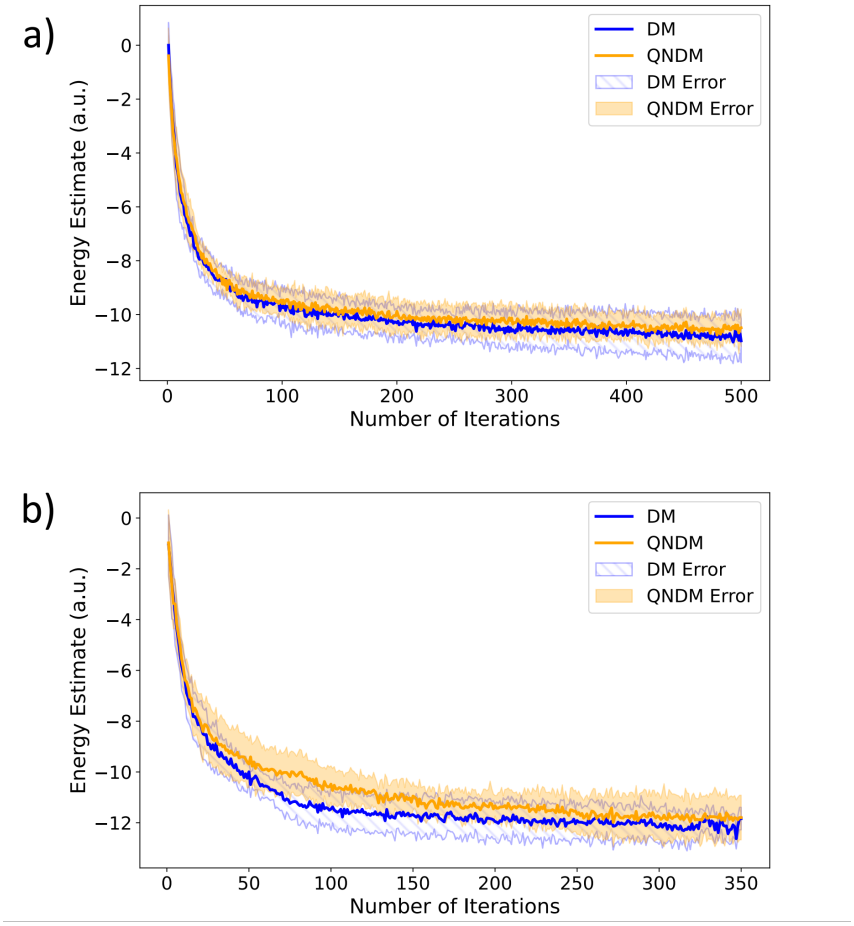


Figure 4.4: The convergence to the minimum energy for large Hamiltonians with $J = 1000$ (b) and $J = 750$ (a). We used $n = 10$ qubits for these simulations, with coefficients h_i were taken from a Gaussian distribution $\mathcal{N}(\mu = 1, \sigma = 0.1)$. For QNDM, a coupling constant of $\lambda = 0.01$ and a gradient descent optimizer with a learning rate of $\eta = 0.05$ were employed. With circuit depth $L = 5$, each energy value was averaged over $N_{\text{shots}} = 10^3$ measurements.

Scaling of computing resources: The most important performance measure is how many logical quantum operations it takes to finish the simulation. By changing J, we can see how both strategies work in different types of computers. Two different systems seem to come up naturally:

- $\boldsymbol{k \ll nJ}$ **regime:** when the number of circuit parameters and logical operations k is small relative to the product of qubits and Hamiltonian terms.
- $\boldsymbol{k \gg nJ}$ **regime:** where the number of operations is much greater than the size of the Hamiltonian, and the complexity of the circuit is the most important factor.

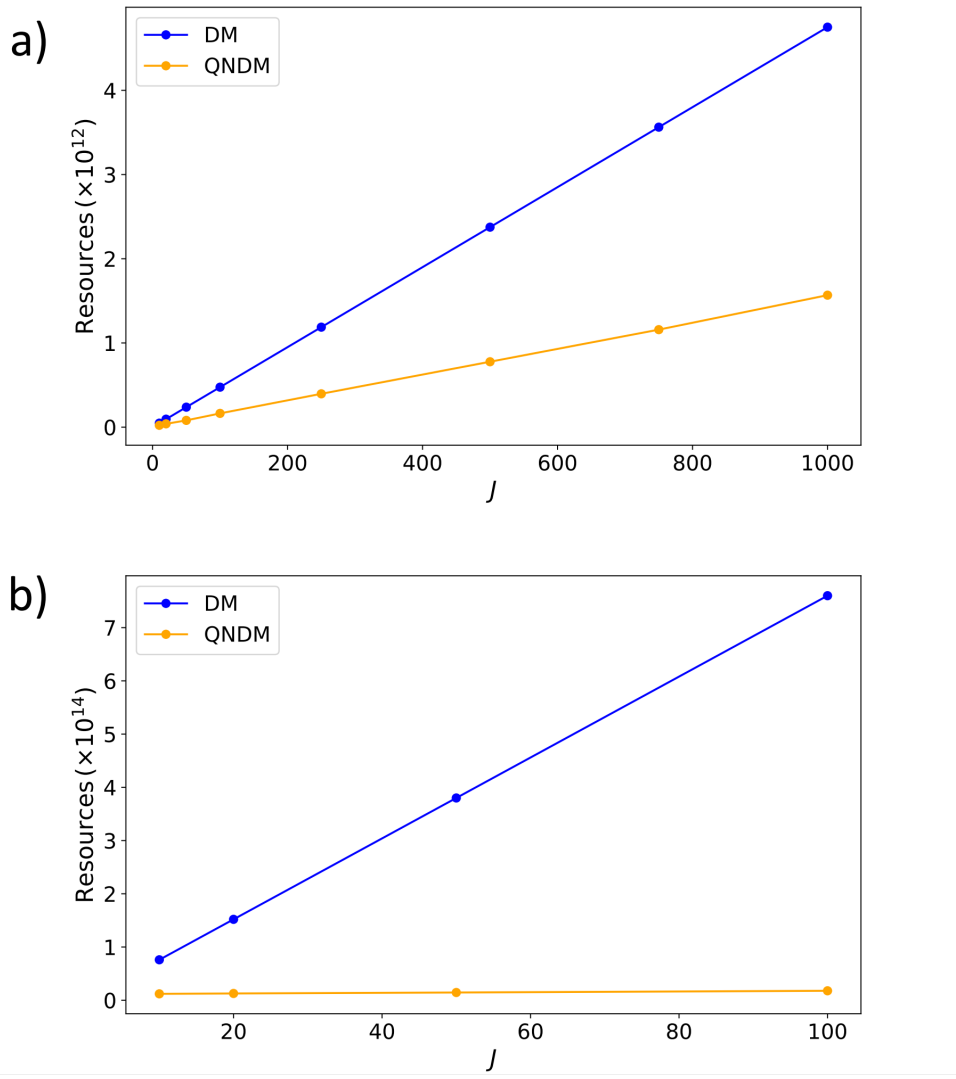


Figure 4.5: Total resource scaling based on J. (a) $k \ll nJ$ regime. (b) $k \gg nJ$ regime. we used $n = 10$ qubits for these simulations. The circuit depth is $L = 5$ for (a), whereas for (b), the same amount of qubits is employed, but with $L = 200$.

Fig. 4.5(a) shows how the total number of logical operations changes as J changes. In this case, QNDM has an obvious edge because it takes a lot fewer steps to attain convergence. For example, when $J = 1000$, the DM technique needs more than four times as many operations as QNDM. This shows that QNDM gets the same results but costs a lot less to run.

High-complexity regime and theoretical expectations: These simulations use a moderate circuit depth of L=5. Increasing the circuit depth could produce more exact energy results, but it would need significantly more computational resources. So, we figured out how the scaling would work in the more difficult case where $k \gg nJ$. Complex physical or chemical systems with several variable parameters are exemplified by this circumstance. When it comes to resource scaling, the QNDM algorithm has a *linear advantage* over the DM approach, according to Solinas *et al.* [171]. This is because QNDM uses a curvature-aware correction term to change the cost-function landscape in a way that makes it easier to find the quantum natural gradient. This enables faster traversal of the parameter landscape with fewer circuit evaluations for each gradient step.

In the predicted outcomes in Fig.4.5(b), we keep J the same but increase the expressivity of the quantum circuit (and hence k). The QNDM technique has a computational cost that stays about the same, needing about 1.7×10^{11} logical operations. The DM technique, on the other hand, scales linearly with J, reaching roughly 7×10^{14} operations. These results agree with the theoretical predictions reported in Ref.[171]. QNDM uses curvature information similar to the natural-gradient framework, which smooths the optimization trajectory and reduces redundant parameter updates. The outcome is a method that converges faster and requires significantly fewer resources, even as problem complexity increases.

The large-Hamiltonian analysis demonstrates that QNDM consistently outperforms DM, even for highly complex quantum systems. Both approaches find the same energy minima, but QNDM always consumes fewer logical quantum operations and converges more smoothly. When $J = 1000$, QNDM cuts the cost of computing by more than four times compared to DM. This reduction becomes even more noticeable when the complexity is great ($k \gg nJ$). As Solinas et al. [171] show, QNDM's use of an approximate quantum Fisher-information metric lets it adaptively scale gradients, which speeds up convergence and cuts down on the number of gates needed. These findings demonstrate that QNDM provides a scalable, accurate, and resource-efficient method for the near future development of quantum hardware and large-scale quantum simulations.

CHAPTER 5

CONCLUSION

5.1 Conclusion

This work provides a detailed analysis of an advanced strategy for enhancing the performance of Variational Quantum Algorithms (VQAs) to find the minimum of the energy in a complex quantum system. These algorithms use both quantum and traditional computing to find the best solution to computationally challenging optimization problems. In these kinds of hybrid models, a quantum computer calculates the system's energy for a specified set of parameters. Then, the outcomes are sent to classical optimizers, which updates the variational parameters to minimize the system energy. After that, the new values are sent back to the quantum system, creating a loop of continual feedback. This process keeps going until the system is at its lowest feasible energy state. These hybrid systems are seen as one of the most useful and promising strategies to link quantum and classical computing. They might soon show a substantial quantum advantage over traditional methods [174].

In this context, two main methods are used to find the derivatives of a cost function, which are essential for optimization. The first strategy is the Direct Measurement (DM) method, which repeatedly measures the quantum system by projective measurements [36, 55, 56]. The second method is the Quantum Non-Demolition Measurement (QNDM) method, which was first suggested in Reference [171]. QNDM employs an auxiliary quantum detector that stores data regarding the gradient and higher-order derivatives of the cost function within its phase, enabling subsequent measurement [172, 171]. The

main difference between the two strategies is how they get and use the information they need.

The QNDM method offers numerous benefits that surpass those of the DM method. The DM approach requires different quantum circuits to measure each derivative, which necessitates numerous runs and measurements. QNDM, on the other hand, can get both the first derivative (gradient) and the higher-order derivatives with just one quantum circuit. In QNDM [166], the quantum detector captures all the relevant data within a single circuit operation without interfering with the system's evolution. The concept is quite similar to the concepts of weak values and weak measurements in quantum physics [170, 169]. This unique feature lets QNDM measure things without changing the system's state, which makes it more efficient and dependable than DM.

Even though QNDM circuits are a little more complicated since they need to store information in the detector phase, they don't have as many logical operations overall. When the quantum operator is made up of numerous Pauli strings, this benefit is further clearer because QNDM processes them all at once without adding to the number of measurements [171, 172]. Another significant benefit of QNDM is that its statistical uncertainty, or mean square error, is independent of the number of Pauli strings (J) included in the observable. In the DM method, on the other hand, the uncertainty goes up in a straight line with J. This means that the DM technique has to run the circuit many more times to have the same level of accuracy as the system gets more complicated and has more Pauli strings. On the other hand, QNDM saves time and resources by maintaining consistent accuracy without requiring additional measurements [172]hh.

In this thesis, we conducted a detailed analysis and comparison of the performances of the QNDM and DM approaches. For our test models, we examined several of the most common molecules that can be simulated on contemporary quantum computers using the IBM Qiskit framework [186]. These simulations supported the analytical results and demonstrated that QNDM always needed fewer computing resources than DM, even for small systems and basic quantum circuits. When working on bigger and more useful challenges, such as simulating chemical compounds and molecular structures [192], this benefit is likely to become much bigger. These findings demonstrate that QNDM is a dependable and effective method for putting Variational Quantum Algorithms into practice on noisy quantum computers, which frequently have constrained resources and are susceptible to errors.

Both DM and QNDM were evaluated on molecular systems including hydrogen (H_2),

dilithium (Li_2), and lithium hydride (LiH) molecules. The molecular information came from the Hamlib database [199] and was run directly on quantum computers. The results showed that both methodologies reached the lowest energy state after about the same level of statistical uncertainty and with about the same number of repetitions. However, in terms of computational resources, QNDM offers a clear advantage. The QNDM approach costs around 70% less to fully optimize the H_2 molecule, 71% less to fully optimize the Li_2 molecule, and about 79% less to fully optimize the LiH molecule. These results show clearly that QNDM works well in real-world quantum simulations.

When the Hamiltonian is complicated, meaning it has a lot of Pauli strings, we find a substantial advantage of the QNDM approach. The conditions considered here are defined in Chap. 3, Sec. (3.6) and (3.7). Our goal is to understand how the computational cost changes as the Hamiltonian's complexity rises. The QNDM approach always needs less computer power than the direct measurement (DM) method. To perform DM on a Hamiltonian with $J = 1000$ Pauli strings, approximately 7×10^{14} logical operations are required. On the other hand, QNDM requires roughly 1.7×10^{11} logical operations. This shows that the QNDM method is far better than the DM method, especially when dealing with Hamiltonians that include a large number of terms.

The results of this thesis demonstrate that Quantum Non-Demolition Measurement (QNDM) provides an efficient framework for variational quantum optimization. The method enables accurate ground-state energy estimation and scalable optimization of complex quantum cost functions. Because of its efficiency it can find applications in quantum chemistry, physics, and optimization problems.

As quantum systems get more complicated, the benefits of QNDM are likely to become even more important. Therefore, QNDM is a practical and innovative approach to developing the next generation of applications for quantum computing.

BIBLIOGRAPHY

[1] L. K. Galla, V. S. Koganti, and N. Nuthalapati, "Implementation of rsa," in *2016 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT)*, pp. 81–87, IEEE, 2016.

[2] T. M. Forcer, A. J. Hey, D. Ross, and P. G. Smith, "Superposition, entanglement and quantum computation," *Quantum Information and Computation*, vol. 2, no. 2, pp. 97–116, 2002.

[3] A. Di Meglio, K. Jansen, I. Tavernelli, C. Alexandrou, S. Arunachalam, C. W. Bauer, K. Borras, S. Carrazza, A. Crippa, V. Croft, *et al.*, "Quantum computing for high-energy physics: State of the art and challenges," *PRX Quantum*, vol. 5, no. 3, p. 037001, 2024.

[4] D. Herman, C. Googin, X. Liu, A. Galda, I. Safro, Y. Sun, M. Pistoia, and Y. Alexeev, "A survey of quantum computing for finance," *arXiv preprint arXiv:2201.02773*, 2022.

[5] M. A. Nielsen and I. L. Chuang, *Quantum computation and quantum information*. Cambridge university press, 2010.

[6] C.-R. Wie, "Two-qubit bloch sphere," *Physics*, vol. 2, no. 3, pp. 383–396, 2020.

[7] U. of Trieste, "Quantum computation iii lecture notes," 2023. Available online.

[8] S. Mangini, "Variational quantum algorithms for machine learning: theory and applications," *arXiv preprint arXiv:2306.09984*, 2023.

[9] H.-Y. Huang, K. Bharti, and P. Rebentrost, "Near-term quantum algorithms for linear systems of equations," 2019.

[10] G. García-Pérez, M. A. Rossi, B. Sokolov, F. Tacchino, P. K. Barkoutsos, G. Mazzola, I. Tavernelli, and S. Maniscalco, "Learning to measure: Adaptive informationally complete generalized measurements for quantum algorithms," *PRX Quantum*, vol. 2, p. 040342, Nov 2021.

[11] K. Bharti, A. Cervera-Lierta, T. H. Kyaw, T. Haug, S. Alperin-Lea, A. Anand, M. Degroote, H. Heimonen, J. S. Kottmann, T. Menke, W.-K. Mok, S. Sim, L.-C. Kwek, and A. Aspuru-Guzik, "Noisy intermediate-scale quantum algorithms," *Rev. Mod. Phys.*, vol. 94, p. 015004, Feb 2022.

[12] V. P. Gerdt, R. Kragler, and A. N. Prokopenya, "On computer algebra application to simulation of quantum computation," in *Proceedings of the UNISA-JINR Symposium "Models and Methods in Few-and Many-Body Systems"(Skukuza, Kruger National Park, South Africa, 6-9 February 2007), to appear*, 2007.

[13] A. Bensoussan, G. Jahangirova, and M. R. Mousavi, "A taxonomy of real faults in hybrid quantum-classical architectures," *arXiv preprint arXiv:2502.08739*, 2025.

[14] M. AbuGhanem and H. Eleuch, "Full quantum tomography study of google's sycamore gate on ibm's quantum computers," *EPJ Quantum Technology*, vol. 11, no. 1, p. 36, 2024.

[15] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. Brandao, D. A. Buell, *et al.*, "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, no. 7779, pp. 505–510, 2019.

[16] R. Pandey, P. Maurya, G. D. Singh, and M. S. Faiyaz, "Simulating quantum principles: Qiskit versus cirq," in *Quantum Computing: A Shift from Bits to Qubits*, pp. 333–348, Springer, 2023.

[17] V. Stirbu, O. Kinanen, M. Haghparast, and T. Mikkonen, "Qubernetes: Towards a unified cloud-native execution platform for hybrid classic-quantum computing," *Information and Software Technology*, vol. 175, p. 107529, 2024.

[18] K. Wright, K. M. Beck, S. Debnath, J. Amini, Y. Nam, N. Grzesiak, J.-S. Chen, N. Pisenti, M. Chmielewski, C. Collins, *et al.*, "Benchmarking an 11-qubit quantum computer," *Nature communications*, vol. 10, no. 1, p. 5464, 2019.

[19] K. Mølmer and A. Sørensen, "Multiparticle entanglement of hot trapped ions," *Physical Review Letters*, vol. 82, no. 9, p. 1835, 1999.

[20] S. Heng, D. Kim, S. Heng, and Y. Han, "Decomposition analysis of quantum native gates on various quantum computers," in *2022 37th International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC)*, pp. 1–3, IEEE, 2022.

[21] IBM Quantum, 2024. URL: https://quantum.ibm.com/.

[22] IonQ, 2024. URL: https://ionq.com/.

[23] IBM Quantum, 2021. URL: https://www.ibm.com/quantum/blog/127-qubit-quantum-processor-eagle.

[24] IonQ, 2024. URL: https://ionq.com/resources/ionq-aria-past-and-future.

[25] J. Preskill, "Quantum computing in the nisq era and beyond," *Bulletin of the American Physical Society*, vol. 64, p. 9, 2019.

[26] Amazon Web Services, 2025. URL: https://aws.amazon.com/braket/quantum-computers/.

[27] F. Tacchino, A. Chiesa, S. Carretta, and D. Gerace, "Quantum computers as universal quantum simulators: state-of-the-art and perspectives," *Advanced Quantum Technologies*, vol. 3, no. 3, p. 1900052, 2020.

[28] Quantinuum, 2025. URL: https://www.quantinuum.com/.

[29] L. Henriet, L. Beguin, A. Signoles, T. Lahaye, A. Browaeys, G.-O. Reymond, and C. Jurczak, "Quantum computing with neutral atoms," *Quantum*, vol. 4, p. 327, 2020.

[30] QuEra Computing Inc., 2025. URL: https://www.quera.com/.

[31] Quandela, 2025. URL: https://www.quandela.com/.

[32] Xanadu Quantum Technologies Inc., 2025. URL: https://www.xanadu.ai/.

[33] J. Degrave, F. Felici, J. Buchli, M. Neunert, B. Tracey, F. Carpanese, T. Ewalds, R. Hafner, A. Abdolmaleki, D. de Las Casas, *et al.*, "Magnetic control of tokamak plasmas through deep reinforcement learning," *Nature*, vol. 602, no. 7897, pp. 414–419, 2022.

[34] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[35] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[36] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio, *et al.*, "Variational quantum algorithms," *Nature Reviews Physics*, vol. 3, no. 9, pp. 625–644, 2021.

[37] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'brien, "A variational eigenvalue solver on a photonic quantum processor," *Nature communications*, vol. 5, no. 1, p. 4213, 2014.

[38] J. Romero, R. Babbush, J. R. McClean, C. Hempel, P. J. Love, and A. Aspuru-Guzik, "Strategies for quantum computing molecular energies using the unitary coupled cluster ansatz," *Quantum Science and Technology*, vol. 4, no. 1, p. 014008, 2018.

[39] E. Farhi, J. Goldstone, and S. Gutmann, "A quantum approximate optimization algorithm," *arXiv preprint arXiv:1411.4028*, 2014.

[40] S. Hadfield, Z. Wang, B. O'gorman, E. G. Rieffel, D. Venturelli, and R. Biswas, "From the quantum approximate optimization algorithm to a quantum alternating operator ansatz," *Algorithms*, vol. 12, no. 2, p. 34, 2019.

[41] S. Lloyd, "Quantum approximate optimization is computationally universal," *arXiv preprint arXiv:1812.11075*, 2018.

[42] M. E. Morales, J. D. Biamonte, and Z. Zimborás, "On the universality of the quantum approximate optimization algorithm," *Quantum Information Processing*, vol. 19, pp. 1–26, 2020.

[43] D. Wecker, M. B. Hastings, and M. Troyer, "Progress towards practical quantum variational algorithms," *Physical Review A*, vol. 92, no. 4, p. 042303, 2015.

[44] H. L. Tang, V. Shkolnikov, G. S. Barron, H. R. Grimsley, N. J. Mayhall, E. Barnes, and S. E. Economou, "qubit-adapt-vqe: An adaptive algorithm for constructing hardware-efficient ansätze on a quantum processor," *PRX Quantum*, vol. 2, no. 2, p. 020310, 2021.

[45] A. Kandala, A. Mezzacapo, K. Temme, M. Takita, M. Brink, J. M. Chow, and J. M. Gambetta, "Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets," *nature*, vol. 549, no. 7671, pp. 242–246, 2017.

[46] J. R. McClean, S. Boixo, V. N. Smelyanskiy, R. Babbush, and H. Neven, "Barren plateaus in quantum neural network training landscapes," *Nature communications*, vol. 9, no. 1, p. 4812, 2018.

[47] M. Cerezo, A. Sone, T. Volkoff, L. Cincio, and P. J. Coles, "Cost function dependent barren plateaus in shallow parametrized quantum circuits," *Nature communications*, vol. 12, no. 1, p. 1791, 2021.

[48] T. Hubregtsen, J. Pichlmeier, P. Stecher, and K. Bertels, "Evaluation of parameterized quantum circuits: on the relation between classification accuracy, expressibility, and entangling capability," *Quantum Machine Intelligence*, vol. 3, no. 1, p. 9, 2021.

[49] M. Ballarin, S. Mangini, S. Montangero, C. Macchiavello, and R. Mengoni, "Entanglement entropy production in quantum neural networks," *Quantum*, vol. 7, p. 1023, 2023.

[50] S. Sim, P. D. Johnson, and A. Aspuru-Guzik, "Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms," *Advanced Quantum Technologies*, vol. 2, no. 12, p. 1900070, 2019.

[51] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[52] T.-D. Guo, Y. Liu, and C.-Y. Han, "An overview of stochastic quasi-newton methods for large-scale machine learning," *Journal of the Operations Research Society of China*, vol. 11, no. 2, pp. 245–275, 2023.

[53] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[54] C. Mohan and K. Deep, *Optimization techniques*. New Age Science, 2009.

[55] M. Schuld, V. Bergholm, C. Gogolin, J. Izaac, and N. Killoran, "Evaluating analytic gradients on quantum hardware," *Physical Review A*, vol. 99, no. 3, p. 032331, 2019.

[56] A. Mari, T. R. Bromley, and N. Killoran, "Estimating the gradient and higher-order derivatives on quantum hardware," *Physical Review A*, vol. 103, no. 1, p. 012405, 2021.

[57] K. Mitarai, M. Negoro, M. Kitagawa, and K. Fujii, "Quantum circuit learning," *Physical Review A*, vol. 98, no. 3, p. 032309, 2018.

[58] D. Wierichs, J. Izaac, C. Wang, and C. Y.-Y. Lin, "General parameter-shift rules for quantum gradients," *Quantum*, vol. 6, p. 677, 2022.

[59] M. Cerezo and P. J. Coles, "Higher order derivatives of quantum neural networks with barren plateaus," *Quantum Science and Technology*, vol. 6, no. 3, p. 035006, 2021.

[60] G. E. Crooks, "Gradients of parameterized quantum gates using the parameter-shift rule and gate decomposition," *arXiv preprint arXiv:1905.13311*, 2019.

[61] L. Banchi, J. Pereira, and S. Pirandola, "Generalization in quantum machine learning: A quantum information standpoint," *PRX Quantum*, vol. 2, no. 4, p. 040321, 2021.

[62] J. Tilly, H. Chen, S. Cao, D. Picozzi, K. Setia, Y. Li, E. Grant, L. Wossnig, I. Rungger, G. H. Booth, *et al.*, "The variational quantum eigensolver: a review of methods and best practices," *Physics Reports*, vol. 986, pp. 1–128, 2022.

[63] P. E. Gill, W. Murray, and M. H. Wright, *Practical optimization*. SIAM, 2019.

[64] J. Liu, H. Yuan, X.-M. Lu, and X. Wang, "Quantum fisher information matrix and multiparameter estimation," *Journal of Physics A: Mathematical and Theoretical*, vol. 53, no. 2, p. 023001, 2020.

[65] S.-I. Amari, "Natural gradient works efficiently in learning," *Neural computation*, vol. 10, no. 2, pp. 251–276, 1998.

[66] J. Stokes, J. Izaac, N. Killoran, and G. Carleo, "Quantum natural gradient," *Quantum*, vol. 4, p. 269, 2020.

[67] A. Arrasmith, M. Cerezo, P. Czarnik, L. Cincio, and P. J. Coles, "Effect of barren plateaus on gradient-free optimization," *Quantum*, vol. 5, p. 558, 2021.

[68] M. J. Powell, "An efficient method for finding the minimum of a function of several variables without calculating derivatives," *The computer journal*, vol. 7, no. 2, pp. 155–162, 1964.

[69] J. A. Nelder and R. Mead, "A simplex method for function minimization," *The computer journal*, vol. 7, no. 4, pp. 308–313, 1965.

[70] M. J. Powell, "Direct search algorithms for optimization calculations," *Acta numerica*, vol. 7, pp. 287–336, 1998.

[71] V. Havlíček, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow, and J. M. Gambetta, "Supervised learning with quantum-enhanced feature spaces," *Nature*, vol. 567, no. 7747, pp. 209–212, 2019.

[72] C. Hempel, C. Maier, J. Romero, J. McClean, T. Monz, H. Shen, P. Jurcevic, B. P. Lanyon, P. Love, R. Babbush, *et al.*, "Quantum chemistry calculations on a trapped-ion quantum simulator," *Physical Review X*, vol. 8, no. 3, p. 031022, 2018.

[73] C. Kokail, C. Maier, R. van Bijnen, T. Brydges, M. K. Joshi, P. Jurcevic, C. A. Muschik, P. Silvi, R. Blatt, C. F. Roos, *et al.*, "Self-verifying variational quantum simulation of lattice models," *Nature*, vol. 569, no. 7756, pp. 355–360, 2019.

[74] Z. Udvarnoki, G. Fáth, and N. Fogarasi, "Quantum advantage of monte carlo option pricing," *Journal of Physics Communications*, vol. 7, no. 5, p. 055001, 2023.

[75] S. Chakrabarti, R. Krishnakumar, G. Mazzola, N. Stamatopoulos, S. Woerner, and W. J. Zeng, "A threshold for quantum advantage in derivative pricing," *Quantum*, vol. 5, p. 463, 2021.

[76] J. R. McClean, J. Romero, R. Babbush, and A. Aspuru-Guzik, "The theory of variational hybrid quantum-classical algorithms," *New Journal of Physics*, vol. 18, no. 2, p. 023023, 2016.

[77] L. D. Landau and E. M. Lifshitz, *Quantum mechanics: non-relativistic theory*, vol. 3. Elsevier, 2013.

[78] D. Griffiths, *Introduction to elementary particles*. John Wiley & Sons, 2020.

[79] I. M. Georgescu, S. Ashhab, and F. Nori, "Quantum simulation," *Reviews of Modern Physics*, vol. 86, no. 1, pp. 153–185, 2014.

[80] E. Anschuetz, J. Olson, A. Aspuru-Guzik, and Y. Cao, "Variational quantum factoring," in *Quantum Technology and Optimization Problems: First International Workshop, QTOP 2019, Munich, Germany, March 18, 2019, Proceedings 1*, pp. 74–85, Springer, 2019.

[81] X. Xu, J. Sun, S. Endo, Y. Li, S. C. Benjamin, and X. Yuan, "Variational algorithms for linear algebra," *Science Bulletin*, vol. 66, no. 21, pp. 2181–2188, 2021.

[82] M. Lubasch, J. Joo, P. Moinier, M. Kiffner, and D. Jaksch, "Variational quantum algorithms for nonlinear problems," *Physical Review A*, vol. 101, no. 1, p. 010301, 2020.

[83] J. Wang and R. Jaiswal, "Scalable quantum ground state preparation of the heisenberg model: A variational quantum eigensolver approach," *arXiv preprint arXiv:2308.12020*, 2023.

[84] S. McArdle, S. Endo, A. Aspuru-Guzik, S. C. Benjamin, and X. Yuan, "Quantum computational chemistry," *Reviews of Modern Physics*, vol. 92, no. 1, p. 015003, 2020.

[85] Y. Cao, J. Romero, J. P. Olson, M. Degroote, P. D. Johnson, M. Kieferová, I. D. Kivlichan, T. Menke, B. Peropadre, N. P. Sawaya, *et al.*, "Quantum chemistry in the age of quantum computing," *Chemical reviews*, vol. 119, no. 19, pp. 10856–10915, 2019.

[86] T. Banks, *Quantum mechanics: an introduction*. CRC Press, 2018.

[87] H. Natori and T. Munehisa, "Time dependent schrödinger equations with vector potentials-numerical calculations and visualizations," *Journal of the Physical Society of Japan*, vol. 66, no. 2, pp. 351–359, 1997.

[88] A. Cowtan, S. Dilkes, R. Duncan, W. Simmons, and S. Sivarajah, "Phase gadget synthesis for shallow circuits," *arXiv preprint arXiv:1906.01734*, 2019.

[89] K. Ender, R. ter Hoeven, B. E. Niehoff, M. Drieb-Schön, and W. Lechner, "Parity quantum optimization: Compiler," *Quantum*, vol. 7, p. 950, 2023.

[90] S. B. Bravyi and A. Y. Kitaev, "Fermionic quantum computation," *Annals of Physics*, vol. 298, no. 1, pp. 210–226, 2002.

[91] J. R. McClean, N. C. Rubin, K. J. Sung, I. D. Kivlichan, X. Bonet-Monroig, Y. Cao, C. Dai, E. S. Fried, C. Gidney, B. Gimby, *et al.*, "Openfermion: the electronic structure package for quantum computers," *Quantum Science and Technology*, vol. 5, no. 3, p. 034014, 2020.

[92] X.-Y. Huang, L. Yu, X. Lu, Y. Yang, D.-S. Li, C.-W. Wu, W. Wu, and P.-X. Chen, "Qubitization of bosons," *arXiv preprint arXiv:2105.12563*, 2021.

[93] T. Hartke, B. Oreg, C. Turnbaugh, N. Jia, and M. Zwierlein, "Direct observation of nonlocal fermion pairing in an attractive fermi-hubbard gas," *Science*, vol. 381, no. 6653, pp. 82–86, 2023.

[94] Z. Zhang, L. Chen, K.-X. Yao, and C. Chin, "Transition from an atomic to a molecular bose–einstein condensate," *Nature*, vol. 592, no. 7856, pp. 708–711, 2021.

[95] X. Liu, J. Li, K. Watanabe, T. Taniguchi, J. Hone, B. I. Halperin, P. Kim, and C. R. Dean, "Crossover between strongly coupled and weakly coupled exciton superfluids," *Science*, vol. 375, no. 6577, pp. 205–209, 2022.

[96] L. Funcke, T. Hartung, K. Jansen, S. Kühn, M. Schneider, P. Stornati, and X. Wang, "Towards quantum simulations in particle physics and beyond on noisy intermediate-scale quantum devices," *Philosophical Transactions of the Royal Society A*, vol. 380, no. 2216, p. 20210062, 2022.

[97] D.-B. Zhang, H. Xing, H. Yan, E. Wang, and S.-L. Zhu, "Selected topics of quantum computing for nuclear physics," *Chinese Physics B*, vol. 30, no. 2, p. 020306, 2021.

[98] L. Nagano, A. Bapat, and C. W. Bauer, "Quench dynamics of the schwinger model via variational quantum algorithms," *Physical Review D*, vol. 108, no. 3, p. 034501, 2023.

[99] E. A. Martinez, C. A. Muschik, P. Schindler, D. Nigg, A. Erhard, M. Heyl, P. Hauke, M. Dalmonte, T. Monz, P. Zoller, *et al.*, "Real-time dynamics of lattice gauge theories with a few-qubit quantum computer," *Nature*, vol. 534, no. 7608, pp. 516–519, 2016.

[100] S. P. Jordan, K. S. Lee, and J. Preskill, "Quantum algorithms for quantum field theories," *Science*, vol. 336, no. 6085, pp. 1130–1133, 2012.

[101] J. Liu, F. Tacchino, J. R. Glick, L. Jiang, and A. Mezzacapo, "Representation learning via quantum neural tangent kernels," *PRX Quantum*, vol. 3, no. 3, p. 030323, 2022.

[102] L. Funcke, T. Hartung, K. Jansen, and S. Kühn, "Review on quantum computing for lattice field theory," *arXiv preprint arXiv:2302.00467*, 2023.

[103] Y. Y. Atas, J. Zhang, R. Lewis, A. Jahanpour, J. F. Haase, and C. A. Muschik, "Su (2) hadrons on a quantum computer via a variational approach," *Nature communications*, vol. 12, no. 1, p. 6499, 2021.

[104] H.-H. Lu, N. Klco, J. M. Lukens, T. D. Morris, A. Bansal, A. Ekström, G. Hagen, T. Papenbrock, A. M. Weiner, M. J. Savage, *et al.*, "Simulations of subatomic many-body physics on a quantum frequency processor," *Physical Review A*, vol. 100, no. 1, p. 012320, 2019.

[105] E. F. Dumitrescu, A. J. McCaskey, G. Hagen, G. R. Jansen, T. D. Morris, T. Papenbrock, R. C. Pooser, D. J. Dean, and P. Lougovski, "Cloud quantum computing of an atomic nucleus," *Physical review letters*, vol. 120, no. 21, p. 210501, 2018.

[106] D. Marcos, P. Rabl, E. Rico, and P. Zoller, "Superconducting circuits for quantum simulation of dynamical gauge fields," *Physical review letters*, vol. 111, no. 11, p. 110504, 2013.

[107] D. Marcos, P. Widmer, E. Rico, M. Hafezi, P. Rabl, U.-J. Wiese, and P. Zoller, "Two-dimensional lattice gauge theories with superconducting quantum circuits," *Annals of physics*, vol. 351, pp. 634–654, 2014.

[108] R. Blatt and C. F. Roos, "Quantum simulations with trapped ions," *Nature Physics*, vol. 8, no. 4, pp. 277–284, 2012.

[109] Z. Davoudi, M. Hafezi, C. Monroe, G. Pagano, A. Seif, and A. Shaw, "Towards analog quantum simulations of lattice gauge theories with trapped ions," *Physical Review Research*, vol. 2, no. 2, p. 023015, 2020.

[110] D. Paulson, L. Dellantonio, J. F. Haase, A. Celi, A. Kan, A. Jena, C. Kokail, R. Van Bijnen, K. Jansen, P. Zoller, *et al.*, "Simulating 2d effects in lattice gauge theories on a quantum computer," *PRX quantum*, vol. 2, no. 3, p. 030334, 2021.

[111] S. Endo, I. Kurata, and Y. O. Nakagawa, "Calculation of the green's function on near-term quantum computers," *Physical Review Research*, vol. 2, no. 3, p. 033281, 2020.

[112] C. Mishra, S. Thompson, R. Pooser, and G. Siopsis, "Quantum computation of an interacting fermionic model," *Quantum Science and Technology*, vol. 5, no. 3, p. 035010, 2020.

[113] J. J. Martinez de Lejarza, L. Cieri, and G. Rodrigo, "Quantum clustering and jet reconstruction at the lhc," *Physical Review D*, vol. 106, no. 3, p. 036021, 2022.

[114] D. Pires, Y. Omar, and J. Seixas, "Adiabatic quantum algorithm for multijet clustering in high energy physics," *Physics Letters B*, vol. 843, p. 138000, 2023.

[115] A. Y. Wei, P. Naik, A. W. Harrow, and J. Thaler, "Quantum algorithms for jet clustering," *Physical Review D*, vol. 101, no. 9, p. 094015, 2020.

[116] G. García-Pérez, M. A. Rossi, B. Sokolov, F. Tacchino, P. K. Barkoutsos, G. Mazzola, I. Tavernelli, and S. Maniscalco, "Learning to measure: Adaptive informationally complete generalized measurements for quantum algorithms," *Prx quantum*, vol. 2, no. 4, p. 040342, 2021.

[117] C. W. Bauer and D. M. Grabowska, "Efficient representation for simulating u (1) gauge theories on digital quantum computers at all values of the coupling," *Physical Review D*, vol. 107, no. 3, p. L031503, 2023.

[118] B. Nachman, D. Provasoli, W. A. De Jong, and C. W. Bauer, "Quantum algorithm for high energy physics simulations," *Physical review letters*, vol. 126, no. 6, p. 062001, 2021.

[119] G. Agliardi, M. Grossi, M. Pellen, and E. Prati, "Quantum integration of elementary particle processes," *Physics Letters B*, vol. 832, p. 137228, 2022.

[120] S. Herbert, "Quantum monte carlo integration: The full advantage in minimal circuit depth," *Quantum*, vol. 6, p. 823, 2022.

[121] J. J. M. de Lejarza, M. Grossi, L. Cieri, and G. Rodrigo, "Quantum fourier iterative amplitude estimation," in *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*, vol. 1, pp. 571–579, IEEE, 2023.

[122] G. Clemente, A. Crippa, K. Jansen, S. Ramírez-Uribe, A. E. Rentería-Olivo, G. Rodrigo, G. F. Sborlini, and L. Vale Silva, "Variational quantum eigensolver for causal loop feynman diagrams and directed acyclic graphs," *Physical Review D*, vol. 108, no. 9, p. 096035, 2023.

[123] S. Ramírez-Uribe, A. E. Rentería-Olivo, G. Rodrigo, G. F. Sborlini, and L. V. Silva, "Quantum algorithm for feynman loop integrals," *Journal of High Energy Physics*, vol. 2022, no. 5, pp. 1–32, 2022.

[124] J. Y. Araz and M. Spannowsky, "Classical versus quantum: Comparing tensor-network-based quantum circuits on large hadron collider data," *Physical Review A*, vol. 106, no. 6, p. 062423, 2022.

[125] A. Blance and M. Spannowsky, "Quantum machine learning for particle physics using a variational quantum classifier," *Journal of High Energy Physics*, vol. 2021, no. 2, pp. 1–20, 2021.

[126] W. Guan, G. Perdue, A. Pesah, M. Schuld, K. Terashi, S. Vallecorsa, and J.-R. Vlimant, "Quantum machine learning in high energy physics," *Machine Learning: Science and Technology*, vol. 2, no. 1, p. 011003, 2021.

[127] S. L. Wu, S. Sun, W. Guan, C. Zhou, J. Chan, C. L. Cheng, T. Pham, Y. Qian, A. Z. Wang, R. Zhang, *et al.*, "Application of quantum machine learning using the quantum kernel algorithm on high energy physics analysis at the lhc," *Physical Review Research*, vol. 3, no. 3, p. 033221, 2021.

[128] A. Blance and M. Spannowsky, "Unsupervised event classification with graphs on classical and photonic quantum computers," *Journal of High Energy Physics*, vol. 2021, no. 8, pp. 1–26, 2021.

[129] C. Tüysüz, F. Carminati, B. Demirköz, D. Dobos, F. Fracas, K. Novotny, K. Potamianos, S. Vallecorsa, and J.-R. Vlimant, "Particle track reconstruction

with quantum algorithms," in *EPJ Web of Conferences*, vol. 245, p. 09013, EDP Sciences, 2020.

[130] A. Gianelle, P. Koppenburg, D. Lucchesi, D. Nicotra, E. Rodrigues, L. Sestini, J. de Vries, and D. Zuliani, "Quantum machine learning for b-jet charge identification," *Journal of High Energy Physics*, vol. 2022, no. 8, pp. 1–24, 2022.

[131] P. Baldi, K. Bauer, C. Eng, P. Sadowski, and D. Whiteson, "Jet substructure classification in high-energy physics with deep neural networks," *Physical Review D*, vol. 93, no. 9, p. 094034, 2016.

[132] J. Lin, M. Freytsis, I. Moult, and B. Nachman, "Boosting $h \to b\bar{b}$ with machine learning," *Journal of High Energy Physics*, vol. 2018, no. 10, pp. 1–25, 2018.

[133] A. collaboration *et al.*, "Identification of jets containing b-hadrons with recurrent neural networks at the atlas experiment," tech. rep., ATL-PHYS-PUB-2017-003, 2017.

[134] T. Felser, M. Trenti, L. Sestini, A. Gianelle, D. Zuliani, D. Lucchesi, and S. Montangero, "Quantum-inspired machine learning on high-energy physics data," *npj Quantum Information*, vol. 7, no. 1, p. 111, 2021.

[135] J. Shlomi, P. Battaglia, and J.-R. Vlimant, "Graph neural networks in particle physics," *Machine Learning: Science and Technology*, vol. 2, no. 2, p. 021001, 2020.

[136] G. Piacquadio and C. Weiser, "A new inclusive secondary vertex algorithm for b-jet tagging in atlas," *Journal of Physics: Conference Series*, vol. 119, no. 3, p. 032032, 2008.

[137] A. Collaboration *et al.*, "Modelling of track reconstruction inside jets with the 2016 atlas$\sqrt{}$ s= 13 tev pp dataset," *CERN, Geneva, Tech. Rep. ATL-PHYS-PUB-2017-016*, 2017.

[138] L. Zhou, S.-T. Wang, S. Choi, H. Pichler, and M. D. Lukin, "Quantum approximate optimization algorithm: Performance, mechanism, and implementation on near-term devices," *Physical Review X*, vol. 10, no. 2, p. 021067, 2020.

[139] M. P. Harrigan, K. J. Sung, M. Neeley, K. J. Satzinger, F. Arute, K. Arya, J. Atalaya, J. C. Bardin, R. Barends, S. Boixo, *et al.*, "Quantum approximate

optimization of non-planar graph problems on a planar superconducting processor," *Nature Physics*, vol. 17, no. 3, pp. 332–336, 2021.

[140] M. Hilbert and P. López, "The world's technological capacity to store, communicate, and compute information," *science*, vol. 332, no. 6025, pp. 60–65, 2011.

[141] P. Wittek, *Quantum machine learning: what quantum computing means to data mining*. Academic Press, 2014.

[142] A. W. Harrow and R. A. Low, "Random quantum circuits are approximate 2-designs," *Communications in Mathematical Physics*, vol. 291, no. 1, pp. 257–302, 2009.

[143] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pp. 212–219, 1996.

[144] R. LaRose and B. Coyle, "Robust data encodings for quantum classifiers," *Physical Review A*, vol. 102, no. 3, p. 032420, 2020.

[145] Y. Liu, S. Arunachalam, and K. Temme, "A rigorous and robust quantum speed-up in supervised machine learning," *Nature Physics*, vol. 17, no. 9, pp. 1013–1017, 2021.

[146] M. Schuld, "Supervised quantum machine learning models are kernel methods," *arXiv preprint arXiv:2101.11020*, 2021.

[147] I. Kerenidis, J. Landman, and A. Luongo, "Anupam prakash qmeans: A quantum algorithm for unsupervised machine learning," *arXiv preprint arXiv:1812.03584*, 2018.

[148] J. S. Otterbach, R. Manenti, N. Alidoust, A. Bestwick, M. Block, B. Bloom, S. Caldwell, N. Didier, E. S. Fried, S. Hong, *et al.*, "Unsupervised machine learning on a hybrid quantum computer," *arXiv preprint arXiv:1712.05771*, 2017.

[149] A. Skolik, S. Jerbi, and V. Dunjko, "Quantum agents in the gym: a variational quantum algorithm for deep q-learning," *Quantum*, vol. 6, p. 720, 2022.

[150] V. Saggio, B. E. Asenbeck, A. Hamann, T. Strömberg, P. Schiansky, V. Dunjko, N. Friis, N. C. Harris, M. Hochberg, D. Englund, *et al.*, "Experimental quantum

speed-up in reinforcement learning agents," *Nature*, vol. 591, no. 7849, pp. 229–233, 2021.

[151] J. A. Cortese and T. M. Braje, "Loading classical data into a quantum computer," *arXiv preprint arXiv:1803.01958*, 2018.

[152] M. Schuld, M. Fingerhuth, and F. Petruccione, "Implementing a distance-based classifier with a quantum interference circuit," *Europhysics Letters*, vol. 119, no. 6, p. 60002, 2017.

[153] M. Weigold, J. Barzen, F. Leymann, and M. Salm, "Expanding data encoding patterns for quantum algorithms," in *2021 IEEE 18th International Conference on Software Architecture Companion (ICSA-C)*, pp. 95–101, IEEE, 2021.

[154] M. Schuld and F. Petruccione, *Machine learning with quantum computers*, vol. 676. Springer, 2021.

[155] E. Grant, M. Benedetti, S. Cao, A. Hallam, J. Lockhart, V. Stojevic, A. G. Green, and S. Severini, "Hierarchical quantum classifiers," *npj Quantum Information*, vol. 4, no. 1, p. 65, 2018.

[156] H.-Y. Huang, M. Broughton, M. Mohseni, R. Babbush, S. Boixo, H. Neven, and J. R. McClean, "Power of data in quantum machine learning," *Nature communications*, vol. 12, no. 1, p. 2631, 2021.

[157] M. Ezawa, "Variational quantum support vector machine based on γ matrix expansion and variational universal-quantum-state generator," *Scientific Reports*, vol. 12, no. 1, p. 6758, 2022.

[158] E. Farhi and H. Neven, "Classification with quantum neural networks on near term processors," *arXiv preprint arXiv:1802.06002*, 2018.

[159] A. Skolik, J. R. McClean, M. Mohseni, P. Van Der Smagt, and M. Leib, "Layerwise learning for quantum neural networks," *Quantum Machine Intelligence*, vol. 3, no. 1, p. 5, 2021.

[160] A. Mari, T. R. Bromley, J. Izaac, M. Schuld, and N. Killoran, "Transfer learning in hybrid classical-quantum neural networks," *Quantum*, vol. 4, p. 340, 2020.

[161] G. Verdon, J. Marks, S. Nanda, S. Leichenauer, and J. Hidary, "Quantum hamiltonian-based models and the variational quantum thermalizer algorithm," *arXiv preprint arXiv:1910.02071*, 2019.

[162] J.-G. Liu and L. Wang, "Differentiable learning of quantum circuit born machines," *Physical Review A*, vol. 98, no. 6, p. 062324, 2018.

[163] P.-L. Dallaire-Demers, M. Stechły, J. F. Gonthier, N. T. Bashige, J. Romero, and Y. Cao, "An application benchmark for fermionic quantum simulations," *arXiv preprint arXiv:2003.01862*, 2020.

[164] B. Coyle, D. Mills, V. Danos, and E. Kashefi, "The born supremacy: quantum advantage and training of an ising born machine," *npj Quantum Information*, vol. 6, no. 1, p. 60, 2020.

[165] M. Nielsen and I. Chuang, *Quantum Computation and Quantum Information*. Cambridge University Press, 10th anniversary edition ed., 2010.

[166] A. A. Clerk, "Full counting statistics of energy fluctuations in a driven quantum resonator," *Physical Review A—Atomic, Molecular, and Optical Physics*, vol. 84, no. 4, p. 043824, 2011.

[167] M. Schuld, V. Bergholm, C. Gogolin, J. Izaac, and N. Killoran, "Evaluating analytic gradients on quantum hardware," *Physical Review A*, vol. 99, no. 3, p. 032331, 2019.

[168] A. Mari, T. R. Bromley, J. Izaac, M. Schuld, and N. Killoran, "Estimating the gradient and higher-order derivatives on quantum hardware," *Physical Review A*, vol. 103, no. 1, p. 012405, 2021.

[169] Y. Aharonov, D. Z. Albert, and L. Vaidman, "How the result of a measurement of a component of the spin of a spin-1/2 particle can turn out to be 100," *Physical review letters*, vol. 60, no. 14, p. 1351, 1988.

[170] A. Bednorz, W. Belzig, and A. Nitzan, "Nonclassical time correlation functions in continuous quantum measurement," *New Journal of Physics*, vol. 14, no. 1, p. 013009, 2012.

[171] P. Solinas, S. Caletti, and G. Minuto, "Quantum gradient evaluation through quantum non-demolition measurements," *The European Physical Journal D*, vol. 77, no. 5, p. 76, 2023.

[172] G. Minuto, D. Melegari, S. Caletti, and P. Solinas, "A novel approach to reduce derivative costs in variational quantum algorithms," *Journal of Physics A: Mathematical and Theoretical*, vol. 58, no. 18, p. 185301, 2025.

[173] L. Banchi and G. E. Crooks, "Measuring analytic gradients of general quantum evolution with the stochastic parameter shift rule," *Quantum*, vol. 5, p. 386, 2021.

[174] J. R. McClean, J. Romero, R. Babbush, and A. Aspuru-Guzik, "The theory of variational hybrid quantum-classical algorithms," *New Journal of Physics*, vol. 18, no. 2, p. 023023, 2016.

[175] A. Arrasmith, L. Cincio, R. D. Somma, and P. J. Coles, "Operator sampling for shot-frugal optimization in variational algorithms," *arXiv preprint arXiv:2004.06252*, 2020.

[176] R. Sweke, F. Wilde, J. Meyer, M. Schuld, P. K. Fährmann, B. Meynard-Piganeau, and J. Eisert, "Stochastic gradient descent for hybrid quantum-classical optimization," *Quantum*, vol. 4, p. 314, 2020.

[177] R. Cheng, "Quantum geometric tensor (fubini-study metric) in simple quantum system: A pedagogical introduction," *arXiv preprint arXiv:1012.1337*, 2010.

[178] P. K. Barkoutsos, F. Gkritsis, P. J. Ollitrault, I. O. Sokolov, S. Woerner, and I. Tavernelli, "Quantum algorithm for alchemical optimization in material design," *Chemical science*, vol. 12, no. 12, pp. 4345–4352, 2021.

[179] P. Solinas and S. Gasparinetti, "Full counting statistics and work fluctuation theorem in quantum systems," *Physical Review E*, vol. 92, no. 3, p. 032136, 2015.

[180] P. Solinas, M. Amico, and N. Zanghì, "Measurement of work and heat in the classical and quantum regimes," *Physical Review A*, vol. 103, no. 6, p. L060202, 2021.

[181] P. Solinas, M. Amico, and N. Zanghì, "Quasiprobabilities of work and heat in an open quantum system," *Physical Review A*, vol. 105, no. 3, p. 032606, 2022.

[182] D. Cox and D. Hinkley, "Theoretical statistics chapman and hall, london," *See Also*, 1974.

[183] J. R. Taylor, *An introduction to error analysis: the study of uncertainties in physical measurements*. MIT Press, 2022.

[184] M. Schuld and N. Killoran, "Quantum machine learning in feature hilbert spaces," *Physical review letters*, vol. 122, no. 4, p. 040504, 2019.

[185] B. Flynn, A. A. Gentile, N. Wiebe, R. Santagati, and A. Laing, "Quantum simulation of the thermodynamics of a frustrated magnet," *New Journal of Physics*, vol. 24, no. 5, p. 053034, 2022.

[186] IBM Quantum, "Qiskit aer: High performance simulators for quantum circuits." https://qiskit.org/ecosystem/aer, 2024. Version 0.14.2 or later.

[187] J. Tilly *et al.*, "The variational quantum eigensolver: a review of methods and best practices," *Physics Reports*, vol. 986, pp. 1–128, 2022.

[188] P. Barkoutsos *et al.*, "Quantum algorithms for electronic structure calculations: A review," *Chemical Reviews*, vol. 121, no. 5, pp. 3061–3120, 2021.

[189] M. Ahookhosh and Y. Nesterov, "An accelerated coordinate gradient method for large-scale smooth optimization," *arXiv preprint arXiv:2107.05958*, 2021.

[190] M. Ahookhosh and Y. Nesterov, "Inexact gradient methods with relative error for smooth convex optimization," *arXiv preprint arXiv:2109.12303*, 2021.

[191] N. Stamatopoulos, G. Mazzola, S. Woerner, and W. J. Zeng, "Towards quantum advantage in financial market risk using quantum generative models," *Quantum*, vol. 6, p. 770, 2022.

[192] S. Weber, A. Chantasri, J. Dressel, A. N. Jordan, K. W. Murch, and I. Siddiqi, "Mapping the optimal route between two quantum states," *Nature*, vol. 511, no. 7511, pp. 570–573, 2014.

[193] T.-C. Chiang and C.-J. Lai, "Molecular ground state simulation by subspace restriction and hund's rule," *arXiv preprint arXiv:2404.03268*, 2024. Available at arXiv:2404.03268 [quant-ph].

[194] A. Szabo and N. Ostlund, "Modern quantum chemistry: Introduction to advanced electronic structure theory, dover publications," *Inc. Mineola, New York*, 1996.

[195] P. W. Atkins and R. S. Friedman, *Molecular quantum mechanics*. Oxford university press, 2011.

[196] T. Helgaker, P. Jorgensen, and J. Olsen, *Molecular electronic-structure theory*. John Wiley & Sons, 2013.

[197] P. J. Knowles and N. C. Handy, "A new determinant-based full configuration interaction method," *Chemical physics letters*, vol. 111, no. 4-5, pp. 315–321, 1984.

[198] A. Javadi-Abhari, M. Treinish, K. Krsulich, C. J. Wood, J. Lishman, J. Gacon, S. Martiel, P. D. Nation, L. S. Bishop, A. W. Cross, *et al.*, "Quantum computing with qiskit," *arXiv preprint arXiv:2405.08810*, 2024.

[199] N. P. Sawaya, D. Marti-Dafcik, Y. Ho, D. P. Tabor, D. E. B. Neira, A. B. Magann, S. Premaratne, P. Dubey, A. Matsuura, N. Bishop, *et al.*, "Hamlib: A library of hamiltonians for benchmarking quantum algorithms and hardware," *Quantum*, vol. 8, p. 1559, 2024.

APPENDIX A

DETECTION WITH A QUBIT

We examine the scenario in which a qubit, representing a two-level quantum system, works as a detector. The initial state of the detector is given by

$$|D_0\rangle = \cos\left(\frac{\alpha}{2}\right)|0\rangle + \sin\left(\frac{\alpha}{2}\right)|1\rangle.$$

The first part of $\mathcal{G}_\lambda$ can be represented as

$$\langle 0|D_0\rangle\langle D_0|1\rangle = \left(\cos\frac{\alpha}{2}\right)\left(\sin\frac{\alpha}{2}\right) = \frac{1}{2}\sin\alpha.$$

The detector gets a phase $\phi(\lambda)$ during the evolution, and the final state is

$$|D_f\rangle = \cos\left(\frac{\alpha}{2}\right)|0\rangle + \sin\left(\frac{\alpha}{2}\right)e^{-i\phi(\lambda)}|1\rangle.$$

Now we find the most prominent part of G_λ:

$$\langle 0|D_f\rangle\langle D_f|1\rangle = \left(\cos\frac{\alpha}{2}\right)\left(\sin\frac{\alpha}{2}e^{i\phi(\lambda)}\right) = \frac{1}{2}\sin\alpha\, e^{i\phi(\lambda)}.$$

So, we can write the function $\mathcal{G}_\lambda$ as follows:

$$\mathcal{G}_\lambda = e^{i\phi(\lambda)}.$$

This indicates directly signifies the phase accrued by the detector throughout its evolution. To see how it works, remember that when $\lambda = 0$, the system and the detector don't interact. So, no phase is built up:

$$[\phi(0) = 0.$$

We find that taking the derivative of G_λ with respect to λ gives us

$$\frac{\partial \mathcal{G}_\lambda}{\partial \lambda} = ie^{i\phi(\lambda)} \frac{\partial \phi(\lambda)}{\partial \lambda}.$$

Making an evaluation at $\lambda = 0$,

$$-i\frac{\partial \mathcal{G}_\lambda}{\partial \lambda}\Big|_{\lambda=0} = \frac{\partial \phi(\lambda)}{\partial \lambda}\Big|_{\lambda=0}.$$

This equation indicates that the average value and the gradient are based on the derivative of the accumulated phase at the origin.

In real experiments, the derivative $\frac{\partial \phi(\lambda)}{\partial \lambda}\big|_{\lambda=0}$ is numerically estimated. You may get close to it by using this formula:

$$\frac{\partial \phi(\lambda)}{\partial \lambda}\Big|_{\lambda=0} \approx \frac{\phi(\Delta\lambda) - \phi(0)}{\Delta\lambda}.$$

So, you can find the physical amount you want by looking at the phase difference between two values of λ that are close together.

APPENDIX B

SECOND DERIVATIVES

In this part, we give a straightforward way to use the parameter-shift rule, Eq. (3.5), to find the second derivative [56]. The usual form of the second derivative is provided as

$$\begin{aligned} g_{j_2,j_1} &= \frac{\partial^2 f(\boldsymbol{\theta})}{\partial\theta_{j_1}\,\partial\theta_{j_2}} \\ &= \Big[f\big(\boldsymbol{\theta}+s(\boldsymbol{e}_{j_1}+\boldsymbol{e}_{j_2})\big) - f\big(\boldsymbol{\theta}+s(-\boldsymbol{e}_{j_1}+\boldsymbol{e}_{j_2})\big) \\ &\quad - f\big(\boldsymbol{\theta}+s(\boldsymbol{e}_{j_1}-\boldsymbol{e}_{j_2})\big) + f\big(\boldsymbol{\theta}+s(-\boldsymbol{e}_{j_1}-\boldsymbol{e}_{j_2})\big)\Big]\,\big[2\sin^2 s\big]^{-1}. \end{aligned} \tag{B.1}$$

We may write Eq.(B.1) in terms of the density matrix that shows the starting n-qubit state, $\rho_s^0 = |\psi_0\rangle\langle\psi_0|$. Then, the expression becomes

$$\begin{aligned} \frac{\partial^2 f(\boldsymbol{\theta})}{\partial\theta_{j_1}\partial\theta_{j_2}} &= \Big[\mathrm{Tr}_S\Big(U^\dagger\big(\boldsymbol{\theta}+s(\boldsymbol{e}_{j_1}+\boldsymbol{e}_{j_2})\big)\,\hat{M}\,U\big(\boldsymbol{\theta}+s(\boldsymbol{e}_{j_1}+\boldsymbol{e}_{j_2})\big)\rho_s^0\Big) \\ &\quad - \mathrm{Tr}_S\Big(U^\dagger\big(\boldsymbol{\theta}+s(-\boldsymbol{e}_{j_1}+\boldsymbol{e}_{j_2})\big)\,\hat{M}\,U\big(\boldsymbol{\theta}+s(-\boldsymbol{e}_{j_1}+\boldsymbol{e}_{j_2})\big)\rho_s^0\Big)\Big]\,[4\sin^2 s]^{-1}. \end{aligned} \tag{B.2}$$

For the Density Matrix (DM) technique, we run the quantum circuit to find the value of each term $f(\boldsymbol{\theta}+s(\boldsymbol{e}_{j_1}+\boldsymbol{e}_{j_2}))$. Then, we do projective measurements for each Pauli string that makes up the observable $\hat{M}$. For the other shifted terms, $f\big(\boldsymbol{\theta}+s(-\boldsymbol{e}_{j_1}+\boldsymbol{e}_{j_2})\big)$, $f\big(\boldsymbol{\theta}+s(\boldsymbol{e}_{j_1}-\boldsymbol{e}_{j_2})\big)$, and $f\big(\boldsymbol{\theta}-s(\boldsymbol{e}_{j_1}+\boldsymbol{e}_{j_2})\big)$, the same steps are used. Figure B.1 shows the circuit that goes with it.

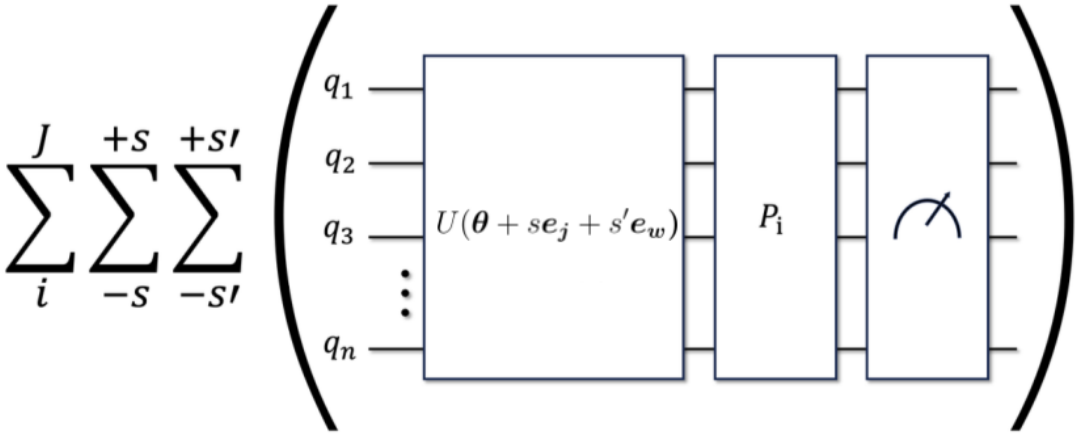


Figure B.1: Quantum circuit of the DM protocol for second derivative calculation. Here, $U(\boldsymbol{\theta} + s\boldsymbol{e}_{j_1} + s'\boldsymbol{e}_{j_2})$ represents the unitary with parameter shifts in both $\boldsymbol{e}_{j_1}$ and $\boldsymbol{e}_{j_2}$ directions. The value J corresponds to the number of Pauli strings, while s and s' are the parameter-shift values. Figure from the Ref.[172].

B.0.1 QNDM Method for the Second Derivative

The QNDM approach lets us find the second derivative without having to go through the circuit for each Pauli string. The whole thing happens in one evolution, as seen in Fig. (B.2).

The total QNDM unitary evolution is

$$U_{\text{tot}}^2 = e^{i\lambda \hat{Z}_a \otimes \hat{M}} U_4 e^{-i\lambda \hat{Z}_a \otimes \hat{M}} U_3 e^{i\lambda \hat{Z}_a \otimes \hat{M}} U_2 e^{-i\lambda \hat{Z}_a \otimes \hat{M}} U_1, \tag{B.3}$$

where the U_y operators (for $y = 1, \ldots, 4$) act in the $\boldsymbol{\theta}$ space as

$$\begin{aligned}
U_1 &= U(\boldsymbol{\theta} + s(\boldsymbol{e}_{j_1} - \boldsymbol{e}_{j_2})), \\
U_2 &= U^\dagger(\boldsymbol{\theta} + s(\boldsymbol{e}_{j_1} - \boldsymbol{e}_{j_2}))U(\boldsymbol{\theta} - s(\boldsymbol{e}_{j_1} + \boldsymbol{e}_{j_2})), \\
U_3 &= U^\dagger(\boldsymbol{\theta} - s(\boldsymbol{e}_{j_1} + \boldsymbol{e}_{j_2}))U(\boldsymbol{\theta} + s(-\boldsymbol{e}_{j_1} + \boldsymbol{e}_{j_2})), \\
U_4 &= U^\dagger(\boldsymbol{\theta} + s(-\boldsymbol{e}_{j_1} + \boldsymbol{e}_{j_2}))U(\boldsymbol{\theta} + s(\boldsymbol{e}_{j_1} + \boldsymbol{e}_{j_2})).
\end{aligned} \tag{B.4}$$

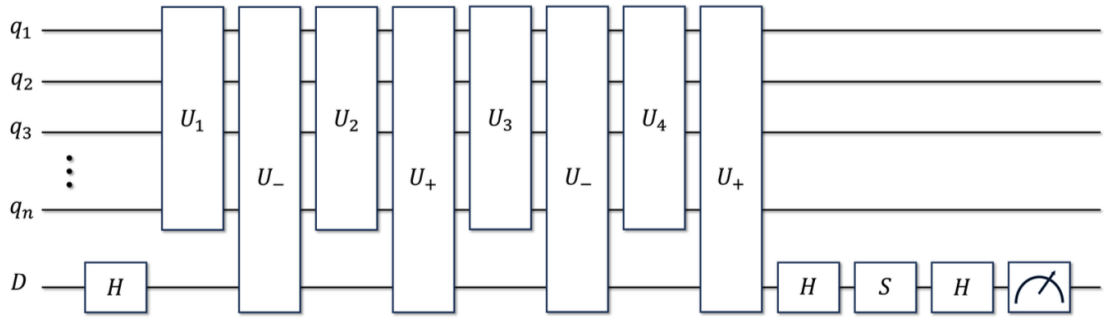


Figure B.2: Quantum circuit for the QNDM protocol. Here, H is the Hadamard gate, and S is the phase gate. The operators are defined as $U_1 = U(\boldsymbol{\theta} + s(\boldsymbol{e}_{j_1} - \boldsymbol{e}_{j_2}))$, $U_2 = U^\dagger(\boldsymbol{\theta} + s(\boldsymbol{e}_{j_1} - \boldsymbol{e}_{j_2}))U(\boldsymbol{\theta} - s(\boldsymbol{e}_{j_1} + \boldsymbol{e}_{j_2}))$, $U_3 = U^\dagger(\boldsymbol{\theta} - s(\boldsymbol{e}_{j_1} + \boldsymbol{e}_{j_2}))U(\boldsymbol{\theta} + s(-\boldsymbol{e}_{j_1} + \boldsymbol{e}_{j_2}))$, and $U_4 = U^\dagger(\boldsymbol{\theta} + s(-\boldsymbol{e}_{j_1} + \boldsymbol{e}_{j_2}))U(\boldsymbol{\theta} + s(\boldsymbol{e}_{j_1} + \boldsymbol{e}_{j_2}))$. The operators $U_\pm = e^{\pm i\lambda \hat{Z}_a \otimes \hat{M}}$ represent the coupling between the system and the detector. Figure from the Ref. [172].

Interpretation

In this case, $\boldsymbol{e}_{j_1}$ and $\boldsymbol{e}_{j_2}$ show two different parameter directions in the vector $\boldsymbol{\theta}$. The second derivative tells us how the function $f(\boldsymbol{\theta})$ changes when both parameters change at the same time. With the DM technique, each shifted configuration needs its own quantum circuit run and measurement. This makes the overall number of evaluations go up. The QNDM method, on the other hand, does all of its measurements in one coherent circuit. This stops the need to run each Pauli term over and over again and lets you get the second derivative straight from the quantum phase that has been added together. Because of this, the QNDM technique is more efficient and cuts down on both the time it takes to do calculations and the number of times a circuit has to be run.

B.1 Error Analysis for Second Derivative

The mean square errors (MSEs) for the second derivatives in both techniques are shown by

$$\mathrm{MSE}_{QNDM}(\hat{g}_{w,j}) = \frac{\lambda^2(\partial_\lambda^2 \mathcal{G}_\lambda)^2}{4} + \frac{\sigma_D^2}{16 N_{QNDM} \sin^4 s\, \lambda^2 (1 - (2P_0 - 1)^2)}. \tag{B.5}$$

The MSE for the density matrix (DM) approach is

$$\mathrm{MSE}_{DM}(\hat{g}_{w,j}) = \sum_i \frac{h_i^2 \sigma_s^2}{4 N_{DM} \sin^4 s}. \tag{B.6}$$

The primary distinction from the first-derivative scenario (see Eqs. (3.34) and (3.38)) is evident solely in the denominators. The MSE_{DM} rises in a straight line with the number

of Pauli strings J for the second derivative. To achieve the same level of accuracy as MSE_{QNDM}, the quantity of measurement shots in the DM method must rise in relation to J. This relationship is shown by

$$\frac{\text{MSE}_{DM}(\hat{g}_{w,j})}{\text{MSE}_{QNDM}(\hat{g}_{w,j})} = 1 \implies N_{DM} = N_{QNDM}\frac{4\sigma_s^2\lambda^2\big(1-(2P_0-1)^2\big)}{\sigma_D^2}\sum_i h_i^2. \tag{B.7}$$

Like Eq.(3.39), we preserve the linear dependency on J, but now there is an extra element that multiplies. This factor takes into account the fact that the MSE formulas for the first and second derivatives have different denominators.

B.2 Cost Simulations for Second Derivative

We examined the asymptotic cost ratios $C^{DM}_{g_{w,j}}/C^{QNDM}_{g_{w,j}}$ for two different scenarios: $k \ll nJ$ (Eq. 3.46) and $k \gg nJ$ (Eq. 3.45). The results are further confirmed by simulations depicted in Figs. B.3 and B.4, which further validate that the same behavior applies to the second derivatives. We took an average across $L = 50$ different realizations for each point in the figures.

Fig. B.3(a) shows how many resources $C^i_{g_{w,j}}$ are needed based on the number of logical operators k, assuming that $k \gg nJ$. The simulations are executed using $n = 10$ qubits, a constant product of $nJ = 300$, and a predetermined number of shots, $N = 200$.

In Fig. B.3(b), we show the ratio of resources needed as the number of Pauli strings J changes for different values of logical operators k in $U(\boldsymbol{\theta})$.

In Fig. B.4(a), we plot the number of resources $C^i_{g_{w,j}}$ against nJ, operating inside the regime where $k \ll nJ$. The simulations are conducted with $n = 10$ qubits, a constant number of logical operators $k = 0.5 \times 10^3$, and a predetermined number of shots $N = 200$.

We demonstrate the ratio of resources as a function of k in Fig. B.4(b) for different values of the product nJ. These results show that the QNDM approach is still better than the DM method, even when finding second derivatives.

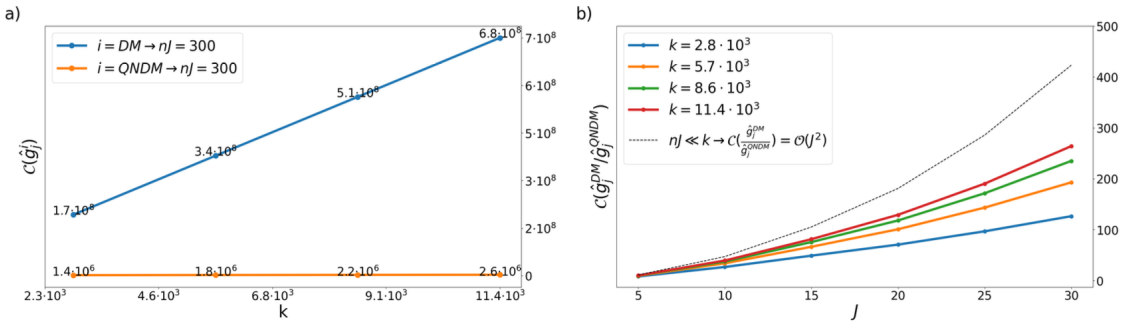


Figure B.3: Regime $k \gg nJ$. (a) The plot shows the number of resources $C(g^i_{w,j})$ required to estimate the second derivative of the cost function along the directions j and w, where $i = QNDM$ (orange line) and $i = DM$ (blue line). The values of $C(g^i_{w,j})$ are shown as a function of the number of logical operators k for a fixed $nJ = 300$. (b) The second panel shows the ratio between DM and QNDM resource numbers, $C^{DM}_{g_{w,j}}/C^{QNDM}_{g_{w,j}}$, as a function of the number of Pauli strings J, where each colored line corresponds to a fixed value of k. The quantum circuit uses $n = 10$ qubits, and all data are averaged over $L = 50$ realizations as discussed in the main text. The number of shots for QNDM is fixed at $N_{QNDM} = 200$, while for the DM method, the number of shots is computed for each realization using Eq. (B.7), so that the MSE errors are the same for both methods. Figure from the Ref. [172].

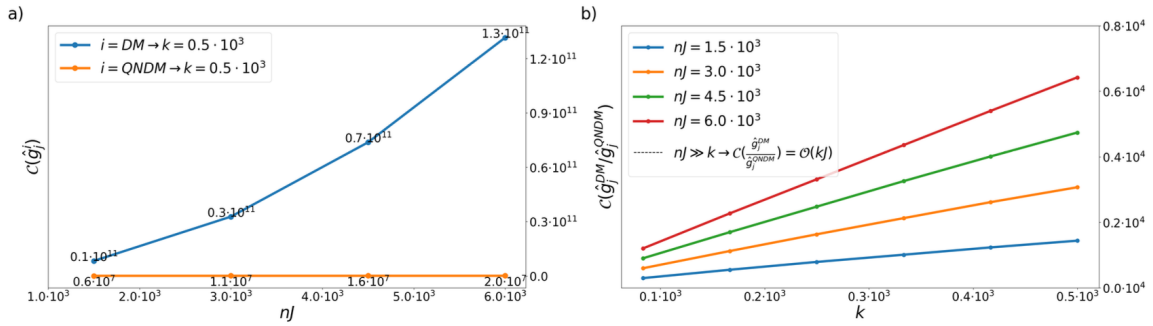


Figure B.4: Regime $k \ll nJ$. (a) The plot shows the averaged number of resources $C(g^i_{w,j})$ required to estimate the second derivative of the cost function along the directions j and w, where $i = QNDM$ (orange line) and $i = DM$ (blue line). The values of $C(g^i_{w,j})$ are shown as a function of the number of Pauli strings J for a fixed number of logical operators $k = 0.5 \times 10^3$. (b) The second panel shows the ratio between DM and QNDM resource numbers, $C^{DM}_{g_{w,j}}/C^{QNDM}_{g_{w,j}}$, as a function of the number of logical operators k, where each colored line corresponds to a fixed value of nJ. The quantum circuit consists of $n = 10$ qubits, and the results are averaged over $L = 50$ realizations as discussed in the main text. The number of shots for QNDM is fixed at $N_{QNDM} = 200$, while for the DM method, the number of shots is calculated for each realization using Eq. (B.7), so that the MSE errors are the same for both methods. Figure from the Ref.[172]