Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

Optimization of Variational Quantum Eigensolvers

Bachelor's Thesis

2024
Michal Švec

Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

Optimization of Variational Quantum Eigensolvers

Bachelor's Thesis

Study Programme:	Computer Science
Field of Study:	Computer Science
Department:	Department of Computer Science
Supervisor:	doc. RNDr. Martin Plesch, PhD.

Bratislava, 2024
Michal Švec

56515481

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta:	Michal Švec
Študijný program:	informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor:	informatika
Typ záverečnej práce:	bakalárska
Jazyk záverečnej práce:	anglický
Sekundárny jazyk:	slovenský

Názov: Optimization of Variational Quantum Eigensolvers
Optimalizácia variačných kvantových eigensolverov

Anotácia: Kvantové počítače zažívajú v posledných rokoch nebývalý rozmach vďaka spoločnosti IBM, ktorá sprístupnila svoje kvantové zariadenia prostredníctvom cloudovej služby. Kvantové počítače, ktoré sú dnes k dispozícii, obsahujú desiatky až niekoľko stoviek qubitov, môžu vykonávať protokol s hĺbkou niekoľkých desiatok krokov a sú silne ovplyvnené chybami a šumom. Vzhľadom na tieto vlastnosti sa často označujú ako NISQ (Noisy intermediate-scale quantum) počítače.
V rámci práce študent získa základné poznatky o základných aspektoch kvantovej mechaniky a fungovaní kvantových počítačov. Zameriame sa na konkrétny hybridný kvantový algoritmus - Variational Quantum Eigensolver. Kombinuje jednoduchú kvantovú úlohu - meranie energie daného stavu - vykonávanú na počítači NISQ s optimalizačnou metódou bežiacou na klasickom zariadení. Študent bude optimalizovať kvantovú časť algoritmu úpravou prípravy stavu pre efektívnejšie využitie kvantových zdrojov.

Vedúci:	doc. RNDr. Martin Plesch, PhD.
Katedra:	FMFI.KI - Katedra informatiky
Vedúci katedry:	prof. RNDr. Martin Škoviera, PhD.
Dátum zadania:	06.11.2023

Dátum schválenia: 06.11.2023

doc. RNDr. Dana Pardubská, CSc.
garant študijného programu

.... študent

.... vedúci práce

56515481

Comenius University Bratislava
Faculty of Mathematics, Physics and Informatics

THESIS ASSIGNMENT

Name and Surname:	Michal Švec
Study programme:	Computer Science (Single degree study, bachelor I. deg., full time form)
Field of Study:	Computer Science
Type of Thesis:	Bachelor´s thesis
Language of Thesis:	English
Secondary language:	Slovak

Title: Optimization of Variational Quantum Eigensolvers

Annotation: Quantum computers have experienced an unprecedented boom in the recent years, thanks to IBM, which has made its quantum devices available using a cloud service. Quantum computers available today contain tens to a few hundred qubits, can execute a protocol with a depth of a few tens of steps, and are heavily influenced by errors and noise. Due to these properties they are often referenced as NISQ (Noisy intermediate-scale quantum) computers.
Within their work the student will gain basic understanding of fundamental aspect of quantum mechanics and working of quantum computers. We will focus on a specific hybrid quantum algorithm – Variational Quantum Eigensolver. It combines a simple quantum task – energy measurement of a given state - performed on a NISQ computer with optimization method running on a classical device. The student will optimize the quantum part of the algorithm by adjusting the preparation of the state for more efficient use of quantum resources.

Supervisor:	doc. RNDr. Martin Plesch, PhD.
Department:	FMFI.KI - Department of Computer Science
Head of department:	prof. RNDr. Martin Škoviera, PhD.
Assigned:	06.11.2023
Approved:	06.11.2023

doc. RNDr. Dana Pardubská, CSc.
Guarantor of Study Programme

.... Student

.... Supervisor

Acknowledgments: I would like to thank my supervisor doc. RNDr. Martin Plesch, PhD. for the opportunity to work on this thesis and for all his advice and time. Undoubtedly, other thanks go to his doctoral students who helped me understand challenging topics and provided me with valuable feedback.

Abstrakt

Dnešné kvantové počítače sa potýkajú s problémami ako chyby, šum a dekoherencia. V súčasnosti kvantové algoritmy samy o sebe nedokážu spoľahlivo riešiť žiadne úlohy. Z toho dôvodu bol uvedený hybridný kvantový algoritmus nazývaný variačný kvantový eigensolver (VQE), ktorý sa snaží tieto problémy zmierniť odovzdaním časti práce klasickým počítačom. V tomto prípade je dôležité zabezpečiť, aby klasické počítače efektívne spolupracovali s kvantovými. Preto sme v našej práci testovali optimalizačné algoritmy a rôzne konfigurácie kvantového počítača s cieľom dosiahnuť čo najlepšie výsledky. Uvažovali sme ideálne kvantové počítače a ako problém, na ktorom sme vykonali všetky naše experimenty, bolo nájdenie základného stavu molekuly vodíka.

Kľúčové slová: kvantové počítače, variačný kvantový eigensolver, ansatz, optimalizačné algoritmy, základný stav energie

Abstract

Quantum computers of today's world face problems such as errors, noise, and decoherence. Currently, quantum algorithms themselves cannot reliably solve any tasks. For this reason, there has been introduced a hybrid quantum algorithm called variational quantum eigensolver (VQE) that attempts to mitigate these problems by delegating part of the work to classical computers. In this case, it is important to ensure efficient collaboration between classical and quantum computers. Hence, in our work, we tested optimization algorithms and various configurations of a quantum computer to achieve the best possible results. We considered ideal quantum computers, and the problem on which we conducted all our experiments was finding the ground state of a hydrogen molecule.

Keywords: quantum computers, variational quantum eigensolver, ansatz, optimization algorithms, ground state energy

Contents

Introduction

In contrast to classical computers, quantum computers are computers that abide by the laws of quantum mechanics, which describes the behavior of particles. Quantum computers promise to solve problems that are intractable for classical computers [1] and make noticeable advancements in many areas. Nevertheless, this is a matter of the future. In this day and age, we work with so-called noisy intermediate-scale quantum (NISQ) computers [2]. NISQ computers are not that big to tackle difficult instances of problems and they suffer from errors, noise, and decoherence.

In this thesis, we deal with a variational quantum eigensolver (VQE) algorithm [1, 3]. The VQE is a hybrid quantum algorithm that combines quantum and classical computing. The quantum part aims to find the energy of a state, given the Hamiltonian (a matrix that describes a quantum system). A classical computer runs an optimization algorithm that attempts to find the best parameters for a parametrized quantum circuit, also called ansatz, that defines the state. This algorithm has many applications, but the most prominent one is finding a ground state energy (a state where electrons are closest to the nucleus of an atom) of molecules.

More specifically, this thesis focuses on the performance of the VQE algorithm. We are particularly interested in how various ansatzes and optimization algorithms can affect the performance of the VQE. Hence, we benchmarked the performance of 18 ansatzes and 15 optimizers. For this purpose, we chose a 4-qubit (quantum bit) representation of a hydrogen molecule. We used VQE from the Qiskit [4] library and ran VQE on a classical computer with a quantum simulator that does not incorporate noise and finite statistics.

The first three chapters primarily cover the theoretical basics of quantum computing and the VQE algorithm. In the fourth chapter, we present observed results from conducted experiments.

Chapter 1

Preliminaries

In this chapter, we will introduce all the necessary concepts that will be used throughout this thesis. It is important to understand these concepts before we proceed further. We will begin with fundamental mathematical concepts, followed by quantum computing principles. Next, we will introduce current quantum computers and the Qiskit library, which can be used for programming quantum computers. Finally, we will discuss Hamiltonians and ground state energy, which are somewhat intertwined with chemistry.

1.1 Mathematics of quantum computing

In the case of standard computers, boolean algebra is used. Quantum computing leverages the power of linear algebra. In this section, we will introduce concepts from linear algebra and some concepts that are more specific to physics. We heavily rely on definitions from the book Mathematical Methods for Physicists by Arfken et al. [5].

1.1.1 Eigenvalues and eigenvectors

Eigenvalues and eigenvectors are some of the most important concepts in linear algebra. The problem of eigenvalues and eigenvectors can be defined by the following equation:

$$A\vec{v} = \lambda\vec{v}, \tag{1.1}$$

where A is a square matrix, vector $\vec{v}$ and constant λ are unknown. If $\vec{v} \neq \vec{0}$, $\vec{v}$ is an eigenvector of matrix A. Each eigenvector has a corresponding eigenvalue λ. Equation 1.1 shows that the resulting vectors after multiplication with matrix A and constant λ are equal. That necessarily means that an eigenvector of a matrix is a vector that does not change its direction when multiplied by that matrix, only its length changes. An eigenvalue is a scalar representing how much the eigenvector is stretched or shrunk. This concept can be easily visually interpreted, see Figure 1.1.

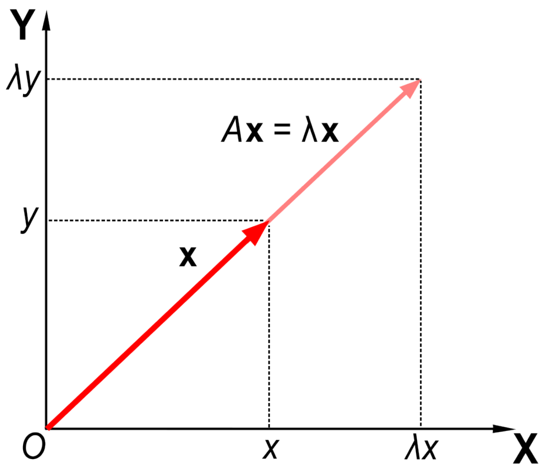


Figure 1.1: Geometric interpretation of eigenvectors and eigenvalues [6]

1.1.2 Complex conjugate

A complex number consists of real and imaginary parts, where the imaginary unit satisfies the equation $i^2 = -1$. If we have a complex number $z = a + bi$, its complex conjugate is the complex number $z^* = a - bi$, where the sign of the imaginary part is flipped.

$$zz^* = (a+bi)(a-bi) = a^2 - (bi)^2 = a^2 + b^2 \tag{1.2}$$

Equation 1.2 reveals that zz^* is a non-negative real number and it enables us to define absolute value as $\sqrt{zz^*}$, which is denoted by $|z|$. A complex conjugate can be also denoted as $\bar{z}$.

1.1.3 Adjoint of a matrix

For matrices with complex elements, a complex conjugate of a matrix is obtained by conjugating all elements of the original matrix. The notation for the complex conjugate of A is A^*. The adjoint of a matrix A, denoted $A^\dagger$ (A dagger), is obtained by both complex conjugating and transposing it. The adjoint of real matrices is just equal to their transpose.

1.1.4 Unitary matrices

Unitary matrices are matrices that satisfy the property $U^\dagger = U^{-1}$, meaning their adjoint equals their inverse. The relationship can be also expressed as follows:

$$UU^\dagger = U^\dagger U. \tag{1.3}$$

Also, provided that U and V are both unitary, then UV and VU will be unitary as well.

1.1.5 Hermitian matrices

The definition of Hermitian matrices builds upon the previous definitions. Hermitian matrices are square matrices that are equal to their adjoint, therefore $H = H^\dagger$. These matrices are also referred to as self-adjoint matrices. All real symmetric matrices are Hermitian. It is important to note that if two matrices A and B are Hermitian, AB or BA will not necessarily be Hermitian. However, it is guaranteed that Hermitian matrices have real eigenvalues.

1.1.6 Pauli matrices

By Pauli matrices, we mean the set of three 2×2 complex matrices. They are defined as follows:

$$\sigma_X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \sigma_Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \sigma_Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{1.4}$$

These matrices are both Hermitian and unitary. Some literature also includes the identity matrix in the set of Pauli matrices.

1.1.7 Tensor product

The tensor product is a widely used operation in quantum computing. This operation applies to several mathematical objects, but in this case, we will restrict ourselves to matrices. The version that works for matrices can be referred to as the Kronecker product. Sometimes are these operations used interchangeably since they use the same notation $\otimes$. Essentially, it is a binary operation that combines two matrices into one larger matrix. Each element of the first matrix is multiplied by the entire second matrix. Mathematically, it is defined as follows:

$$A \otimes B = \begin{pmatrix} a_{11}B & a_{12}B & \dots & a_{1n}B \\ a_{21}B & a_{22}B & \dots & a_{2n}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}B & a_{m2}B & \dots & a_{mn}B \end{pmatrix}. \tag{1.5}$$

1.1.8 Bra-ket notation

Bra-ket notation, also known as Dirac notation, plays an important role in quantum mechanics. It is a notation for vectors used to describe a quantum state. A ket is a standard column vector whereas a bra is an adjoint of a ket.

$$\langle \alpha | = \begin{pmatrix} a_1^* & a_2^* & \dots & a_n^* \end{pmatrix} \qquad |\beta\rangle = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

Bra Ket

The advantage of this notation is that it facilitates the expression vector operations such as inner product:

$$\langle\alpha|\beta\rangle = \begin{pmatrix} a_1^* & a_2^* & \dots & a_n^* \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix} = \sum_{i=1}^{n} a_i^* b_i, \tag{1.6}$$

outer product:

$$|\beta\rangle\langle\alpha| = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix} \begin{pmatrix} a_1^* & a_2^* & \dots & a_n^* \end{pmatrix} = \begin{pmatrix} b_1a_1^* & b_1a_2^* & \dots & b_1a_n^* \\ b_2a_1^* & b_2a_2^* & \dots & b_2a_n^* \\ \vdots & \vdots & \ddots & \vdots \\ b_na_1^* & b_na_2^* & \dots & b_na_n^* \end{pmatrix}, \tag{1.7}$$

and tensor product:

$$|\alpha\rangle \otimes |\beta\rangle = |\alpha\rangle|\beta\rangle = |\alpha\beta\rangle. \tag{1.8}$$

1.1.9 Hilbert space

In simple terms, it is a finite-dimensional complex vector space equipped with an inner product as defined in equation 1.6. Although there are more detailed definitions of Hilbert space that extend to infinite-dimensional vector spaces, we will adhere to this simple definition since our work deals exclusively with finite-dimensional vector spaces.

1.2 Introduction to quantum computing

The standard computers, as we know them, for their functioning use laws of standard mechanics. Quantum computers, on the other hand, use laws of quantum mechanics. Quantum mechanics describes the behavior of particles at the microscopic level, whereas standard mechanics deals with macroscopic objects. The objective of this section is to highlight the most important concepts and provide at least a brief idea of quantum computing. The main source of information for this section is the book Quantum Computation and Quantum Information by Michael A. Nielsen and Isaac L. Chuang [7].

1.2.1 Qubit

A qubit is an abbreviation of a quantum bit. It is a bit counterpart in quantum computing, thus a basic unit of information in quantum computers. Classical bits can hold only two values, either 0 or 1. However, qubits are more complex.

Mathematically, a qubit is represented by a vector in a two-dimensional Hilbert space. Basis vectors of this vector space:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \text{ and } |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \tag{1.9}$$

are also known as computational basis states and they are analogous to classical bits 0 and 1. In addition, qubits can be in a so-called superposition of states. This means that a state of a qubit can be a linear combination of states $|0\rangle$ and $|1\rangle$:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \text{ where } \alpha, \beta \in \mathbb{C} \text{ and } |\alpha|^2 + |\beta|^2 = 1, \tag{1.10}$$

therefore there are infinitely many states that a qubit can be in. In case we consider two qubits, the basis vectors of this four-dimensional Hilbert space are $|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$. Then the superposition looks as follows:

$$|\psi\rangle = \alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle, \tag{1.11}$$

where the sum of the squared coefficients is equal to 1 as well.

All these vectors and complex numbers may be difficult to understand and imagine. For simplification, we can leverage the Bloch sphere to visualize the state of a qubit. It is a unit sphere named after physicist Felix Bloch. For the sake of simplicity, we will not go into the details, but if we use the properties of a quantum state, there is a possibility to rewrite it cleverly, such that a quantum state can be visualized as a vector in the Bloch sphere.

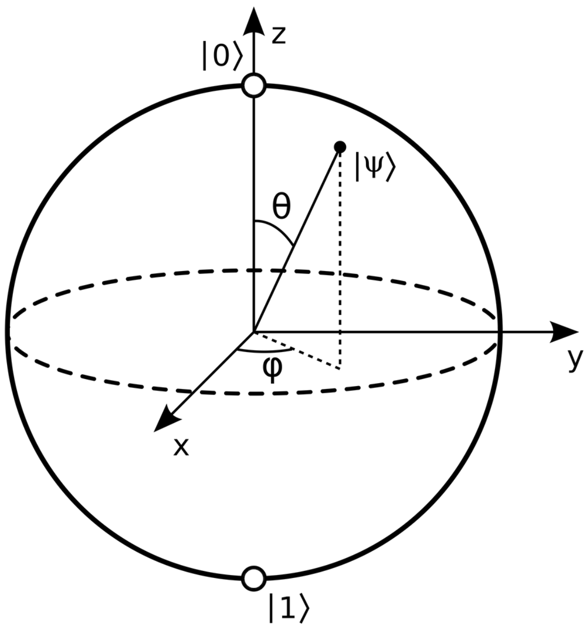


Figure 1.2: Bloch sphere [8]

In the real world, physical qubits can be implemented in different ways. There is a plethora of options but the most prominent ones used by leading companies are superconductors and trapped ions (an atom that is not neutral). Also, we cannot omit photon-based qubits that can succeed as well.

1.2.2 Measurements

A measurement is an operation that enables us to determine the state of a qubit. However, this operation does not work as most of us would expect. When a qubit is measured, it yields either outcome $|0\rangle$, with a probability of $|\alpha|^2$, or outcome $|1\rangle$, with a probability of $|\beta|^2$. We are working with probabilities, so the normalization condition, $|\alpha|^2 + |\beta|^2 = 1$, should make more sense now. A measurement is a destructive operation. Upon the first measurement, the state of a qubit is collapsed to either $|0\rangle$ or $|1\rangle$ and any subsequent measurements will yield the same result. The original state cannot be recovered after the measurement. Table 1.1 shows canonical measurements on the x, y, and z axes, however, there are infinitely many ways to measure a qubit, depending on how we rotate it.

Measurement axis	**States**
x-axis	$\|+\rangle$ and $\|-\rangle$
y-axis	$\|-i\rangle$ and $\|+i\rangle$
z-axis	$\|0\rangle$ and $\|1\rangle$

Table 1.1: Measurements and their respective states

Most (if not all) contemporary quantum computers perform measurements only on the z-axis (computational basis) [9]. Table 1.2 demonstrates how measurements on the x and y axes can be converted to the z-axis measurement.

Measurement axis	**Conversion**
x-axis	rotation of a state around y-axis by -90 degrees
y-axis	rotation of a state around x-axis by 90 degrees

Table 1.2: Measurements and their conversion to computational basis

1.2.3 Quantum gates

Thus far, our focus has been on examining the properties of qubits, leaving the question of qubit manipulation unanswered. This section delves into the fundamental quantum gates, accompanied by visual representations to enhance comprehension. All visualizations were produced using the Qiskit [4] and QuTiP [10] libraries.

Quantum gates are the quantum equivalent of classical logic gates. In contrast to logic gates in classical computing, quantum gates are represented by matrices. The only property that a matrix must adhere to is unitarity. There are infinitely many unitary matrices, therefore we have infinitely many quantum gates.

Another specialty of quantum computation is its reversibility. This means that from the output of a gate, we can always determine the input. This is not the case for logic gates. For instance, the *AND* gate is not reversible. From the output, we cannot determine the input.

X, Y, Z gates

Gates X, Y, and Z are the most fundamental single-qubit gates. All three gates perform rotation of a state by 180 degrees, the X gate around the x-axis, the Y gate around the y-axis, and the Z gate around the z-axis. They are also known as bit-flip, phase-flip, and bit-phase-flip gates, respectively. Parametrized equivalents of these gates are called RX, RY, and RZ. These gates are used to rotate the state of a qubit by a given angle.

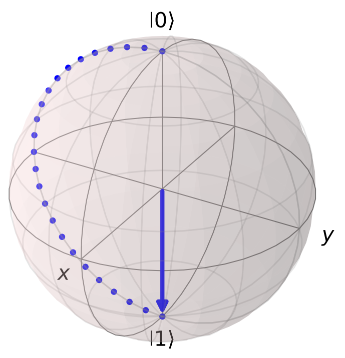
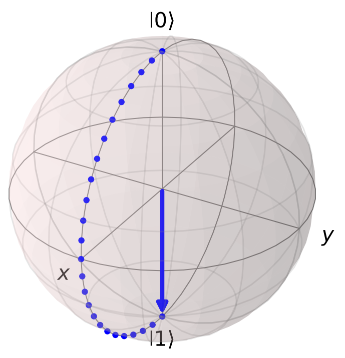
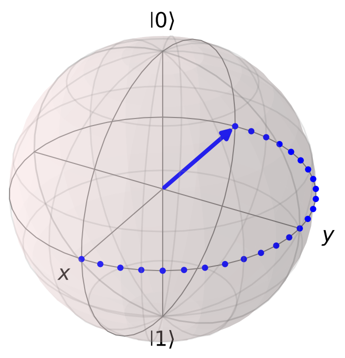
X-gate	**Y-gate**	**Z-gate**
$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$
$-\boxed{X}-$	$-\boxed{Y}-$	$-\boxed{Z}-$
		

Table 1.3: X, Y, and Z gates and their representations

Note that the initial state of the Z-gate differs from the initial state of the X and Y gates. If we had started from the state $|0\rangle$, we would have not seen any change.

Hadamard gate

The Hadamard operation can be thought of as a two-step process. It is a rotation of a state around the y-axis by 90 degrees and then subsequent rotation around the x-axis by 180 degrees. If we apply the Hadamard gate on a qubit in the state $|0\rangle$, we get a qubit in an equal superposition of states $|0\rangle$ and $|1\rangle$.

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

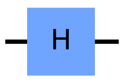


Figure 1.3: Hadamard gate representation

Controlled-NOT gate

The controlled-NOT (CNOT) gate operates on two input qubits, known as the control qubit and the target qubit. The action of the gate may be described as follows. If the control qubit is set to state $|0\rangle$, then the target qubit remains untouched. Conversely, if the control qubit is set to state $|1\rangle$, then the target qubit is flipped.

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

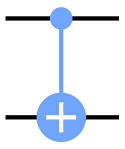


Figure 1.4: CNOT gate representation

1.2.4 Quantum entanglement

Apart from the superposition, quantum entanglement is another quantum phenomenon that gives us an advantage over classical computers. When qubits are entangled, we mean that they are somehow bound together and they are dependent. Altering the state of one qubit will immediately alter the state of the other qubit predictably. In the below example, we will demonstrate the simplest entangled state, also known as the Bell state.

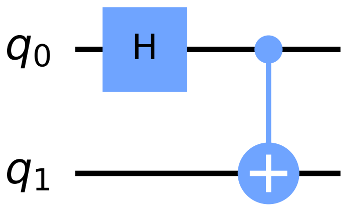


Figure 1.5: Bell state circuit

$$|00\rangle = |0\rangle \otimes |0\rangle \tag{1.12}$$

$$\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes |0\rangle \tag{1.13}$$

$$\frac{1}{\sqrt{2}}(|0\rangle \otimes |0\rangle + |1\rangle \otimes |0\rangle) \tag{1.14}$$

$$\frac{1}{\sqrt{2}}(|0\rangle \otimes |0\rangle + |1\rangle \otimes |1\rangle) \tag{1.15}$$

$$\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \tag{1.16}$$

In the following lines, we explain individual steps of the computation.
(1.12) Initial state of the circuit.

(1.13) Hadamard gate applied, the first qubit is equally likely to be in state $|0\rangle$ or $|1\rangle$.
(1.14) Expanded bracket.
(1.15) CNOT gate applied, the first qubit is $|1\rangle$, state of the second qubit will be flipped.
(1.16) Final state of the circuit.

From the above example, we can see that the qubits are correlated. This particular state is a superposition only of two states $|00\rangle$ and $|11\rangle$, without the entanglement we would have to consider a superposition of four states $|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$. If the first qubit is measured to be in state $|0\rangle$, then the second one will also be in state $|0\rangle$. The same applies to state $|1\rangle$.

1.2.5 Quantum simulation

The introduction above does not explicitly highlight the capabilities of quantum computers. We will try to illustrate the advantage of quantum computers in this simple example from Quantum Computation and Quantum Information book by Michael A. Nielsen and Isaac L. Chuang [7].

Suppose we have a system containing 50 qubits. To describe a state of such a system requires $2^{50} \approx 10^{15}$ complex amplitudes. If the amplitudes are stored in 128 bits of precision, then it requires 256 bits or 32 bytes to store each amplitude. In total 32×10^{15} is about 32 thousand terabytes and this amount hits the limits of current supercomputers. It is important to realize that with each additional qubit, the memory requirements are doubled. Basically, in quantum computers, we can represent states more efficiently.

1.3 Noisy Intermediate-Scale Quantum computers

So far, we have covered the very fundamentals of quantum computing and briefly outlined its advantages. However, in today's world, quantum computers still suffer from several problems, namely the following ones.

Scalability

The more qubits we have, the bigger instances of problems we can solve. However, with an increasing number of qubits and number of gates we introduce errors into quantum computation. The gates themselves, especially the entangling ones, possess a certain probability that the outcome of the gate will result in an error. Also, qubit connectivity goes hand in hand with the scalability. It is not very rational to have many qubits if they are not connected and we cannot use multiqubit gates on them.

Quantum decoherence

Qubits are very sensitive and their state can be easily influenced by the noise from the environment. By noise we mean magnetic fields, radio waves, vibrations, light, and more. To minimize this effect, quantum processing units (QPUs) are accompanied by other components that are used to shield the qubits from the environment and keep them at temperatures close to absolute zero ($0\text{K} = -273.15°\text{C}$). Thanks to that, quantum computers are large in size and they often resemble chandeliers even though QPUs are in size comparable to processors in standard computers.

Lack of error correction

Theoretically, we always consider so-called logical qubits that do not have any problems and work seamlessly. In reality, we use physical qubits that suffer from noise and decoherence and it hinders us from executing quantum algorithms reliably. To mitigate this problem, we can use quantum error correction. The idea is that we can use multiple physical qubits, from tens even to thousands, to create a single reliable logical qubit. By doing so, we run into a problem with scalability.

Quantum computers that match these characteristics are called noisy intermediate-scale quantum (NISQ) computers. These characteristics and the term NISQ were introduced by John Preskill [2] and its meaning is the following. The term "noisy" refers to all the noise that quantum computers currently suffer from. The noise will significantly constrain the capabilities of quantum computers in the foreseeable future. "Intermediate-scale" denotes quantum computers expected to emerge in the coming years, featuring a qubit count ranging from 50 to a few hundred.

1.4 Qiskit

This section describes the solution that we use for working with quantum simulators and quantum algorithms.

Do not confuse programming a quantum computer with standard high-level programming as we know it from classical computers, we are not there yet. The programming of quantum computers is more like programming in assembly language. A thorough knowledge of computer's hardware and architecture is crucial in assembly language programming, as it involves the manipulation of hardware through the use of low-level instructions. A similar principle is applied here, qubits are manipulated using quantum gates. For this purpose, we decided to go with the open-source Qiskit (Quantum Information Science Kit) [4] library for Python backed by IBM.

There are also other alternatives like Cirq (by Google) [11], Pennylane (by Xanadu) [12], Q# (by Microsoft) [13], Sliq (by ETH Zürich) [14], and many more. The reason why we decided on Qiskit is that it serves our purpose and is far ahead of its competitors. Competitors offer nowhere near what Qiskit offers. It is the most popular quantum computing library. It provides plenty of learning resources, tutorials, videos, and as it is open-source, there is a big community around it. IBM has built an entire ecosystem around it [15] with libraries for quantum machine learning, chemistry, finance, and many more. The 7-year work of IBM culminated in the middle of February 2024, when they released version 1.0.0 of Qiskit. Even though Qiskit is mainly developed by IBM, it is not limited to IBM's quantum computers. It can support the hardware of other companies through additional packages.

1.5 Ground state energy

This section draws inspiration from the book Chemistry: The Central Science by Brown et al. [16]. Before delving into the ground state energy, let's revisit an atom first. Atoms are the smallest building blocks of matter. They are composed of 3 subatomic particles, protons, neutrons, and electrons. Protons and neutrons are located within the nucleus of the atom. As their names suggest, protons have a positive electrical charge, while neutrons are electrically neutral. Electrons have a negative electrical charge. Atoms themselves are neutral, so the number of protons must be equal to the number of electrons. Electrons are attracted to the protons in the nucleus by the electrostatic force that exists between particles of the opposite electrical charge.

Electrons are organized into orbitals, each with its own characteristic shape and energy level. An electron has the ability to transition between orbitals by either absorbing or emitting photons with energy precisely matching the difference in energy between the two orbits. In order for the electron to transition to a higher-energy state, it must absorb energy. Conversely, energy is emitted when the electron transitions to a lower-energy state. The lowest-energy state is also known as the ground state and higher-energy states are called excited states. We will measure this energy in the Hartree (Ha) units. Hartree is a unit of energy in the atomic units system [17].

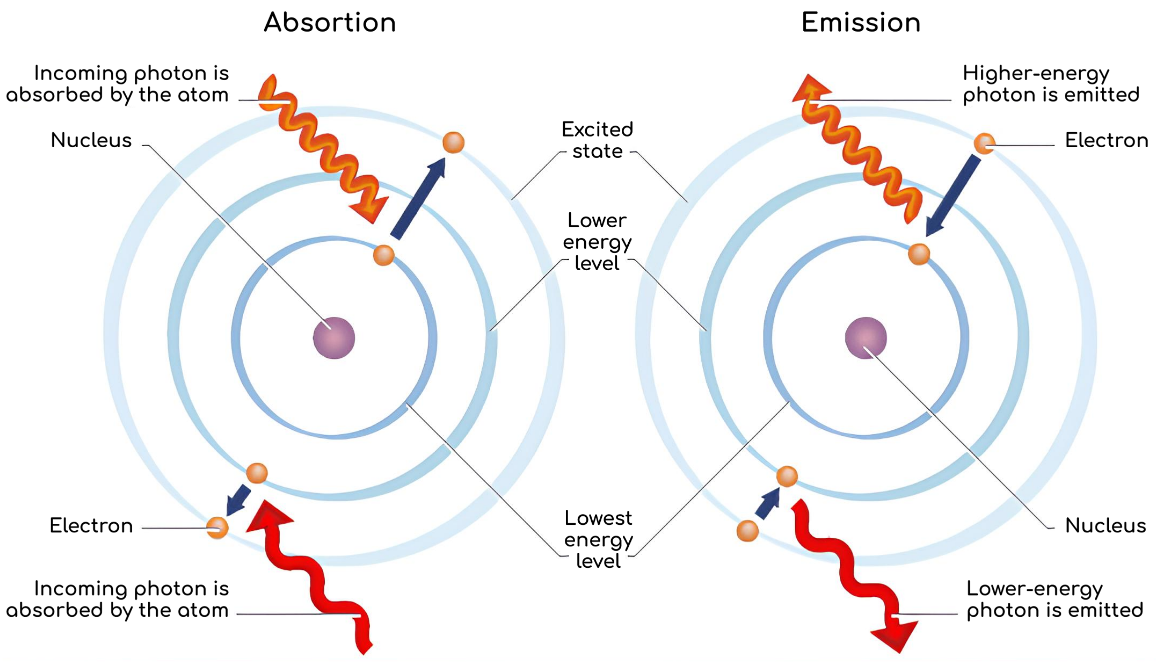

Absorption
Emission
Incoming photon is absorbed by the atom
Nucleus
Excited state
Lower energy level
Lowest energy level
Electron
Incoming photon is absorbed by the atom
Higher-energy photon is emitted
Electron
Nucleus
Lower-energy photon is emitted

Figure 1.6: Absorption and emission of a photon by an electron [18]

1.6 Hamiltonian

The Hamiltonian defines the total energy of a physical system. Many different forms of Hamiltonians exist in physics and chemistry, but for us, it is just a matrix. Once a Hamiltonian is constructed, it must be translated into operators that can be directly measured on a quantum computer. The representation for quantum computers looks as follows [19]:

$$\hat{H} = \sum_{i=1}^{m} c_i \sigma_i,\ \sigma_i \in \{I, X, Y, Z\}^{\otimes n},\ c_i \in \mathbb{R}, \tag{1.17}$$

where I is an identity matrix and X, Y, Z are Pauli matrices which we discussed in section 1.1.

The Pauli matrices represent measurements. For instance, the expression $c_1 Z_0 X_1 Y_2$ means that we measure qubit zero on the z-axis, qubit one on the x-axis, qubit two on the y-axis, and then we will multiply the results together with the coefficient c_1. Sometimes this Hamiltonian representation is referred to as a sum of Pauli strings. A ground state energy is a real number, therefore this Hamiltonian representation has real eigenvalues. This fact concludes that the Hamiltonian is a Hermitian matrix.

Chapter 2

Ansatz

Before we delve into details, let us preface this section with an etymology of the word ansatz. An ansatz is a term borrowed from German (plural ansätze), referring to an educated guess, an initial point, or an additional assumption made to facilitate solving a problem, which may later be verified based on the results obtained [20].

In the context of quantum computing, an ansatz refers to a parametrized quantum circuit, which is comprised of quantum gates and some of them are parametrized. Ansatzes are often used in variational algorithms where the circuit parameters are optimized by classical computers.

In this chapter, we will discuss the role of the ansatz, its properties, types, and introduce several commonly used ansatzes. Moreover, we will address the challenges that can impede the design of an ansatz.

2.1 Expressibility

The notion of expressibility can be very helpful to get an understanding what is the role of an ansatz. Expressibility tells how much of the Hilbert space can be covered by the ansatz. The following figures should provide a clear explanation of this idea.

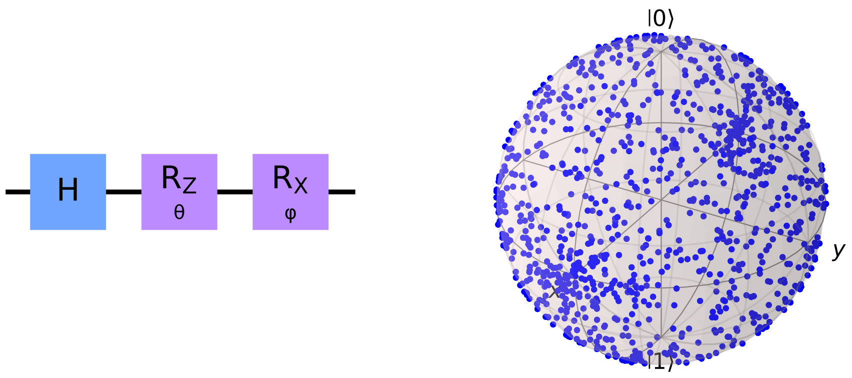


Figure 2.1: Ansatz expressibility, 1000 parameters sampled uniformly randomly

Provided we know that our solution is a real number, we can use a simpler ansatz that covers only the real part of the Hilbert space. Additional gates would introduce more noise into a circuit and the more parameters we have, the more difficult it is to optimize them.

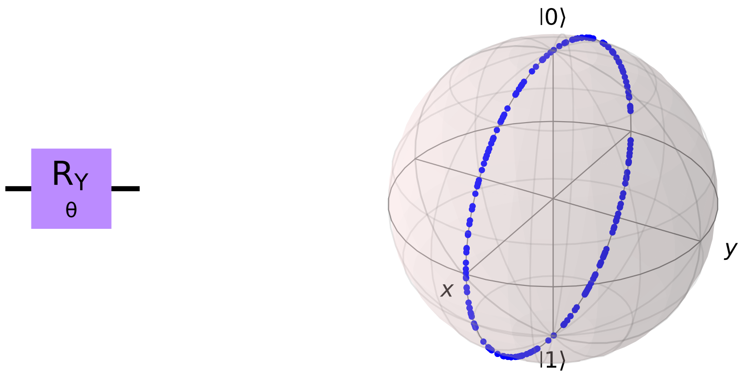


Figure 2.2: Ansatz expressibility, 200 parameters sampled uniformly randomly

2.2 Trainability

The trainability of an ansatz denotes its ability to efficiently optimize its parameters, typically through iterative processes. The major hurdle that can impede the trainability of an ansatz is a barren plateau problem (vast planes in cost function) which we will introduce in the following chapter. Holmes et al. [21] claim that trainability and expressibility are inversely related. Furthermore, they claim that highly expressive ansatzes are more prone to barren plateaus and therefore are harder to train, however, that does not necessarily mean that inexpressive ansatzes cannot have trainability issues. The priority should be to span only the relevant part of the Hilbert space, ideally only where the solution lies and minimize the number of parameters and gates that can introduce noise. Designing an efficient ansatz involves finding an optimal trade-off between expressibility and trainability [21].

2.3 Circuit depth

The depth of a circuit is a measure that expresses how many "levels" of gates a quantum circuit contains. Alternatively, it is the longest path in a circuit. It is a way to increase expressibility but at the same time a way how to introduce more noise into a circuit.

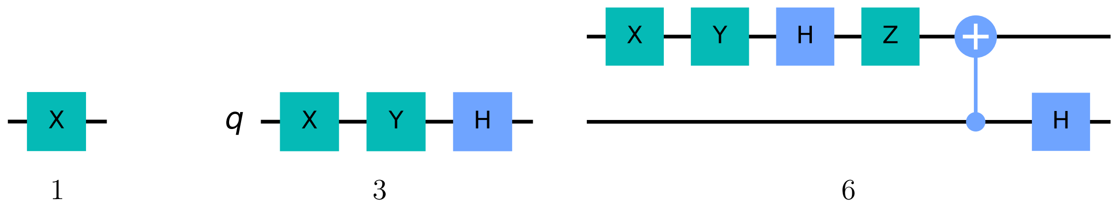


Figure 2.3: Circuit depth

2.4 Hardware efficient ansatz

An ansatz designed with hardware constraints in mind. The goal is to minimize a circuit depth, the number of gates, and the number of entangling gates in such a way it can be easily implemented on real hardware and thereby tries to minimize error, noise, and decoherence. The fact that an ansatz is hardware efficient also depends on the hardware used. This aspect will be briefly addressed in the following subsection.

The structure of hardware efficient ansatz (HEA) consists of layers, each layer consists of rotation gates and entangling gates. A single layer can contain multiple levels of rotations with various gates. The final layer does not contain any entangling gates, only rotation gates.

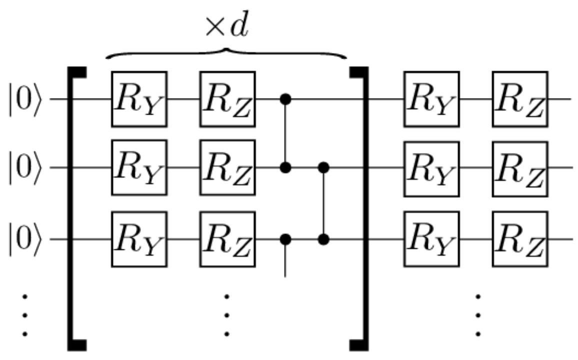


Figure 2.4: Hardware efficient ansatz structure [22]

The main drawback of hardware efficient ansatzes is that they can span a substantial portion of the Hilbert space, often exceeding what is necessary [21] and that can lead to undesirable problems with trainability. There is ongoing research about ansatzes and also there is an article that claims to have trainability guarantees for hardware efficient ansatzes that possess specific properties [23]. Nevertheless, we cannot be certain about that. Other research endeavors attempted to disrupt this theory by introducing a new source of untrainability [24].

Qubit topology and transpilation

To deepen understanding and foster motivation of hardware efficient ansatzes, we will explore specifics of IBM hardware in more detail. IBM has many quantum computers with various qubit counts, topologies, and native gate sets. A qubit topology shows qubit connectivity. Upon observation of the qubit topology depicted in Figure 2.5, it becomes apparent that certain qubits are disconnected and situated far from one another. One might ask how to implement a two-qubit gate between these qubits. The process is outlined in the IBQ documentation [25, 26], but in brief, it can be summarized as follows.

To apply a 2-qubit gate to a quantum circuit between qubits that are not directly connected on a quantum computer, it is necessary to incorporate one or more swap gates into a circuit. These swap gates facilitate the rearrangement of qubit states until they are positioned adjacent to each other on a device gate map. Each swap gate is both a noisy and expensive operation to perform. Determining the minimum number of swap gates needed to align a circuit with a device is crucial. However, finding the optimal swap mapping is challenging as this problem belongs to the NP-hard problems, making it expensive to compute for larger quantum devices. Qiskit defaults to using a stochastic heuristic algorithm to compute a satisfactory swap mapping.

Moreover, besides connectivity issues, compatibility with the hardware's native gate set must be addressed. Each processor family operates with its own set of native gates. By default, these systems only support operations within their respective gate sets. Consequently, every gate in the circuit needs to be translated into elements of this set. This process is called circuit transpilation and Qiskit provides multiple optimization levels. The higher the level is, the more difficult it is to compute that and it may result in a more optimal circuit. Fortunately, it is handled automatically by Qiskit so users do not have to worry about it.

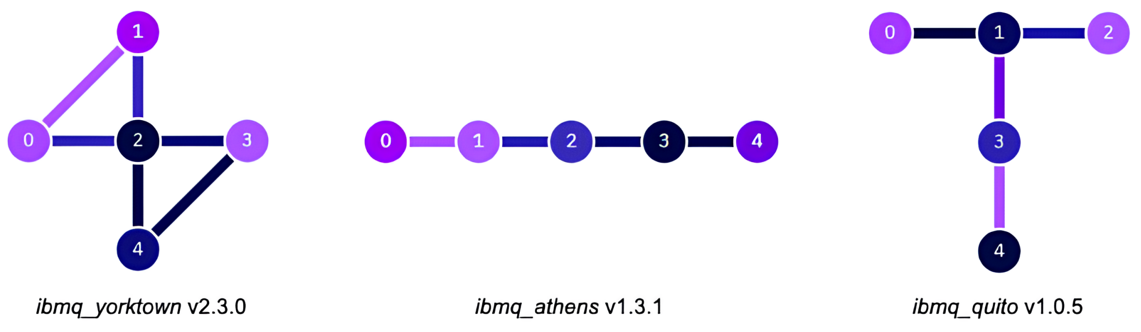


Figure 2.5: Example qubit topology of IBM quantum computers [27]

Alternatives

There are multiple types of ansatzes and broadly can be classified into two categories, problem-inspired and hardware-inspired. In addition to that, if ansatz leverages a structure of a particular problem, is said to be problem-specific/problem-derived, otherwise, it is problem-agnostic/general.

One such example of a problem-inspired ansatz is the Unitary Coupled Cluster (UCC) ansatz proposed for finding the ground state energy of molecules. A circuit depth of this ansatz grows $O(n^4)$, where n is the number of qubits [28]. This is not viable for larger problems and current NISQ devices. Therefore, problem-agnostic hardware efficient approaches are typically employed.

2.4.1 Commonly used ansatzes

This section will introduce several commonly used hardware efficient ansatzes categorized by type of entanglement. All the ansatzes come from IBM Quantum documentation [29].

Linear entanglement

In this ansatz, each qubit is entangled with the next qubit.

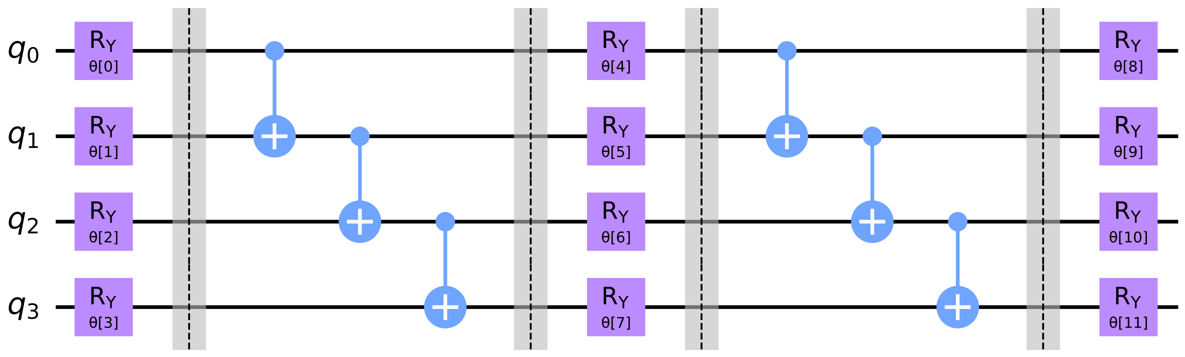


Figure 2.6: Linear entanglement ansatz, 2 layers, RY rotation gates

Reverse linear entanglement

As the name suggests, this is the linear ansatz with a reversed order of entangling gates.

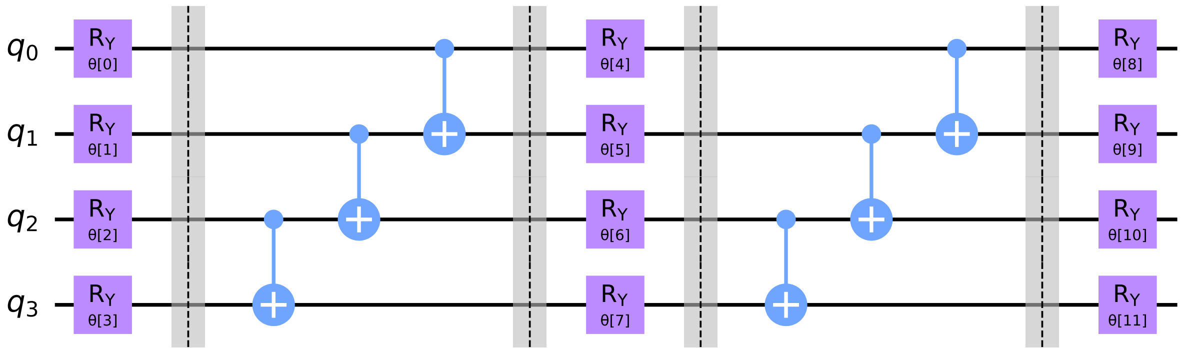


Figure 2.7: Reverse linear entanglement ansatz, 2 layers, RY rotation gates

Pairwise entanglement

The entanglement layer consists of two "levels". In the first level, qubit i is entangled with qubit $i+1$ for all even values of i, and then in the second level qubit i is entangled with qubit $i+1$ for all odd values of i.

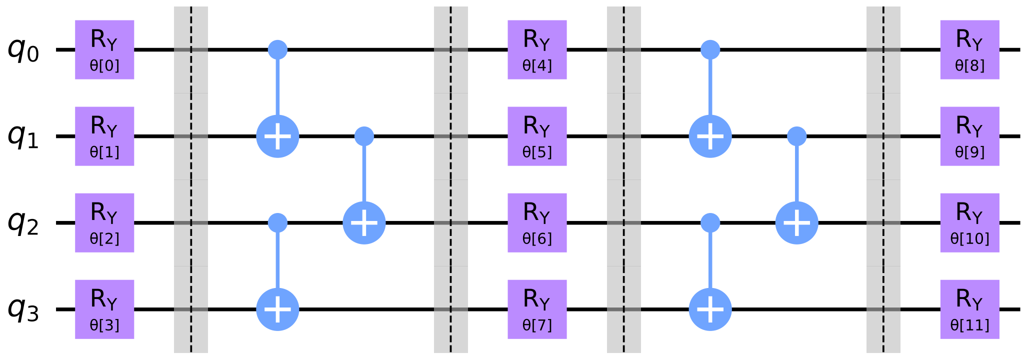


Figure 2.8: Pairwise entanglement ansatz, 2 layers, RY rotation gates

Circular entanglement

This is an extension of the linear entanglement where the last qubit is also entangled with the first qubit. In case physical qubits are arranged into a circle, it is easy to implement.

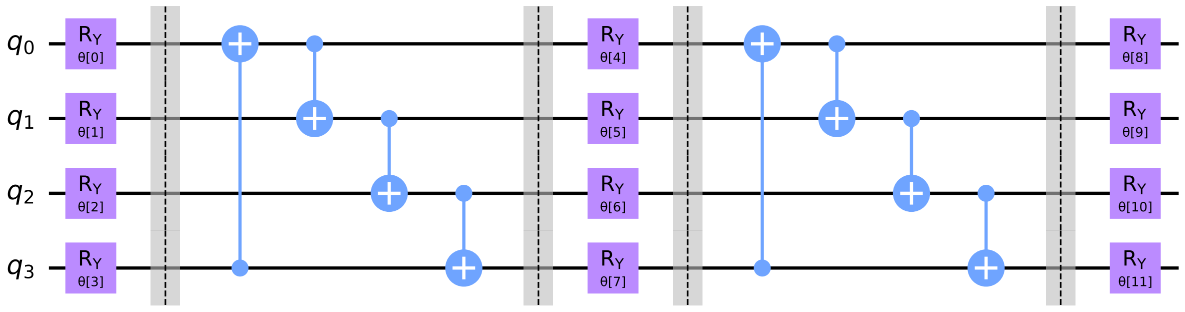


Figure 2.9: Circular entanglement ansatz, 2 layers, RY rotation gates

Shifted circular alternating (SCA) entanglement

SCA entanglement ansatz consists of circular entanglement where the entangling gate connecting the first with the last qubit is shifted by one in every layer. Furthermore, the role of control and target qubits are swapped in every layer (therefore alternating).

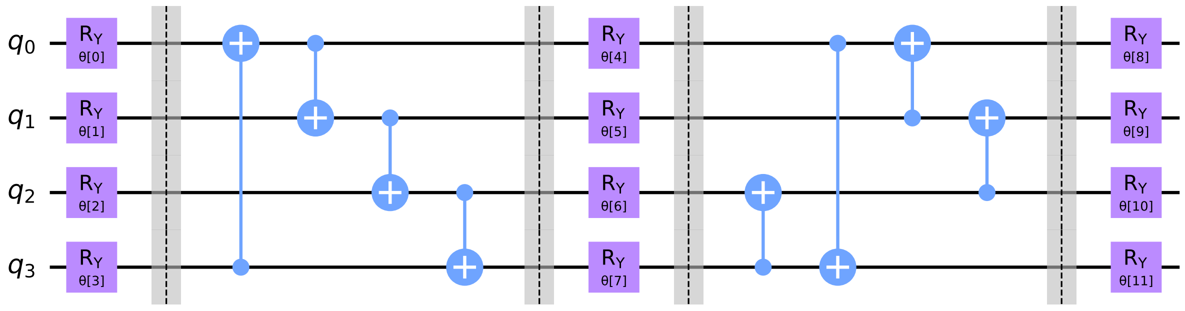


Figure 2.10: Shifted circular alternating entanglement ansatz, 2 layers, RY rotation gates

Full entanglement

This ansatz entangles each qubit with every other qubit. At first glance it may seem that this ansatz cannot be hardware efficient due to the amount of CNOT gates, however, conceptually it is still considered hardware efficient. In fact, the CNOT gates have minimal impact on the resulting state since there are no rotation gates interleaved with the entangling gates. A block containing only CNOT gates is not particularly useful. Moreover, the circuit depth remains manageable. Provided that the reverse linear ansatz uses CNOT gates as entangling gates, this ansatz has the same unitary matrix as the reverse linear ansatz but with more CNOT gates.

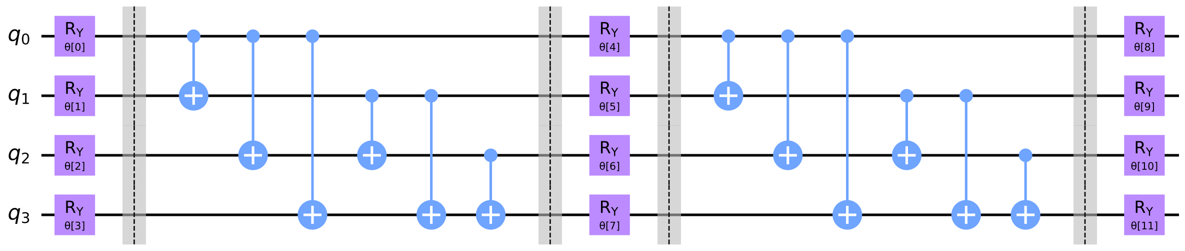


Figure 2.11: Full entanglement ansatz, 2 layers, RY rotation gates

Chapter 3

Variational quantum eigensolver

At first glance, the term variational quantum eigensolver may seem complicated and it does not say anything to people outside of quantum computing. Thus, the objective of this chapter is to provide a clear explanation of the variational quantum eigensolver and clarify why it is termed as it is. For the rest of this thesis, we will use the abbreviation VQE.

The VQE is a hybrid algorithm that attempts to find eigenvalues of a Hamiltonian. The term "hybrid" refers to a scenario where part of the algorithm runs on a quantum computer and part of the algorithm runs on a classical computer. Essentially, it is a loop where quantum and classical computers alternate, until a result is found or a maximum number of iterations is reached.

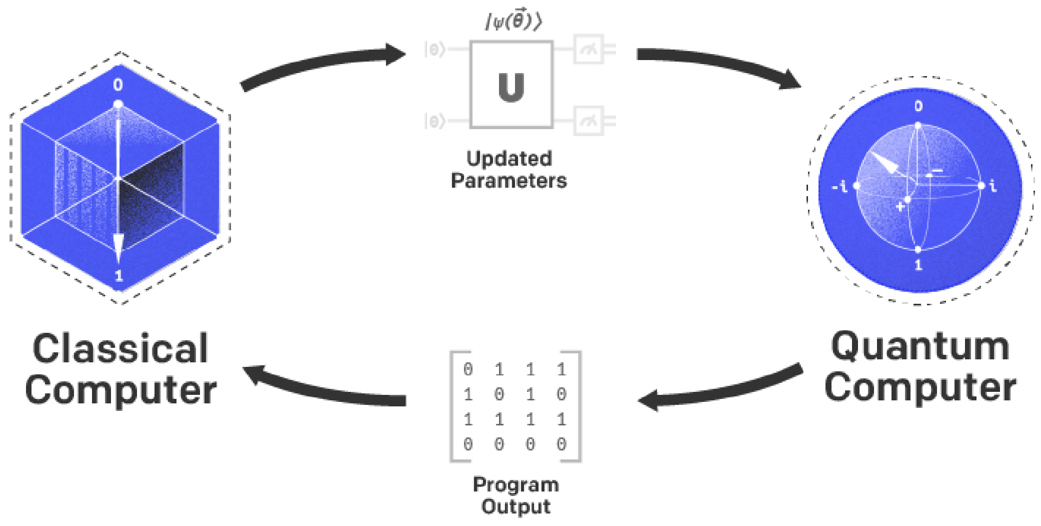


Figure 3.1: Hybrid algorithm that runs on both quantum and classical computers [30]

Apart from the VQE, there are other algorithms that can solve for eigenvalues. One of them is the quantum phase estimation algorithm (QPE). The QPE is purely a quantum algorithm that imposes significant requirements on quantum hardware and that will not be feasible in the near future [2]. This problem led to the introduction of the VQE. The VQE tries to reduce the demand on resources, by shifting part of the work to standard computers [3].

In this thesis, we deal with finding the ground state energy of a molecule. Arguably, this is the most prominent use case, but this is not the only application of the VQE. It can be used for any problem that can be mapped to Hamiltonian expression. For instance, in the area of finance, it can be a portfolio optimization problem [31].

3.1 Variational principle

In quantum mechanics, there is the famous Schrödinger equation ($\hat{H}\Psi = E\Psi$), which is a partial differential equation that describes how the quantum state of a physical system behaves. Solving this equation analytically is very hard, in most cases, we must resort to computers to determine the solutions [32]. This imposes very high time and memory requirements on computers, therefore we rely on the variational method which gives us an approximation of the ground state energy of a quantum system and enables us to solve the problem much more efficiently.

3.2 Eigensolver

As we indicated above, this algorithm solves for eigenvalues of a given Hamiltonian. Specifically, we are interested in the lowest eigenvalue because that eigenvalue corresponds to the ground state energy of a given molecular Hamiltonian.

Let $\hat{H}$ be our Hamiltonian, U denotes a unitary matrix of our ansatz with parameters θ, and $|0\rangle$ is the initial state. The goal of the VQE algorithm is to find the best set of parameters θ that will minimize the cost function. The ground state energy can be computed using the following formula [3]:

$$E = \min_{\theta} \frac{\langle 0|U(\theta)^{\dagger}\hat{H}U(\theta)|0\rangle}{\langle 0|U(\theta)^{\dagger}U(\theta)|0\rangle}. \tag{3.1}$$

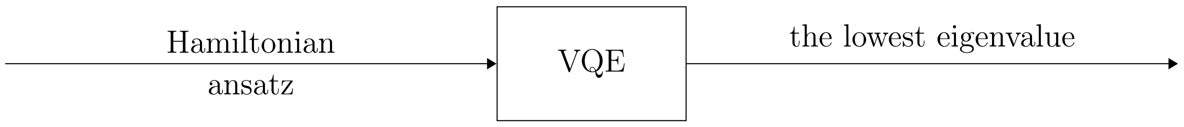


Figure 3.2: VQE input and output scheme

One might question why there is a need for a quantum computer to solve for eigenvalues when a classical computer can do it in polynomial time. The advantage of the VQE follows from the fact that if Hamiltonian is defined on n qubits, that is still $O(n)$ space complexity for quantum computers. For classical computers, it is $O(exp(n))$ space complexity and that can be infeasible.

3.3 Optimization algorithms

Optimization algorithms, in short known as optimizers, are algorithms that take a function as an input and try to find the parameters that can lead to a minimal/maximal value of the function. In our scenario, an optimizer tries to find angles to function 3.1 defined in the previous section. There is a plethora of optimizers and broadly can be divided into two types, gradient-based and gradient-free.

3.3.1 Gradient-based and gradient-free optimization algorithms

Mathematics provides us with a valuable tool called derivatives. In this particular case, we use partial derivatives. Partial derivatives are just derivatives of a multivariable function. We always differentiate just by one variable and we treat other variables as constants. A gradient is a vector of partial derivatives [5]. Suppose we have a multivariable function $f(x_1, x_2, \ldots, x_n)$, then the gradient of this function is defined as follows:

$$\nabla f(x_1, x_2, \ldots, x_n) = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{pmatrix}. \tag{3.2}$$

The idea of gradient-based algorithms is to leverage the gradient to make a step in the direction of the steepest descent. However, some optimization algorithms go beyond the gradient and also use a Hessian matrix. The Hessian matrix is a square matrix of second-order partial derivatives. This method is called Newton's method [33]. If the Hessian matrix is expensive to compute and instead it is approximated, we are talking about the so-called quasi-Newton method [34]. Conversely, algorithms that do not use the gradient are called gradient-free and they use some other methods to make a step in the direction of the steepest descent. Gradient-free algorithms may be useful when the gradient is not available or is expensive to compute.

3.3.2 Barren plateau

Simply put, a barren plateau is a vast flat landscape of a cost function. This leads to the issue with trainability as optimization algorithms have difficulties escaping this area. A common driver of barren plateaus is ansatz expressibility [21]. When using a hardware efficient ansatz there are many parameters and some of the parameters do not have any impact on the final result. Even upon changing the parameters, the result remains the same and this is how the barren plateau problem arises. However, this is not the only source of a barren plateau, there are other sources like system size, random initialization, noise, degree of entanglement, and others [3]. Several strategies have been proposed to avoid or alleviate barren plateaus including customizing an ansatz, employing highly sophisticated parameter initialization techniques, and other advanced methods [3].

Chapter 4

Results

This chapter covers the most important experiments and obtained results for ansatzes and optimizers. Additionally, we provide insights on how we proceeded, on technical aspects, and also some thought processes that influenced our decisions.

Before proceeding further, it is essential to clarify the following terms that will frequently appear in this chapter.

An *iteration* is a single run of the VQE encompassing both quantum and classical parts. The number of iterations determines how many times an ansatz will obtain a new set of optimized parameters.

When an optimization algorithm attempts to find the best parameters for an ansatz, it evaluates a cost function. *Cost function evaluations* count how many times the cost function was evaluated.

4.1 Benchmark setup

Our objective was to find the ground state energy of a hydrogen molecule and figure out some optimizations concerning ansatzes and optimizers. Initially, we started by trying to run the VQE with few different optimizers and ansatzes. Even though the ground state is just a real number, we wanted to know how the energy convergence progresses. We did that by plotting an energy convergence graph similar to that in Figure 4.1. For illustration purposes, we chose two optimizers, *COBYLA* and *Gradient Descent*, and six ansatzes we mentioned in section 2.4.1.

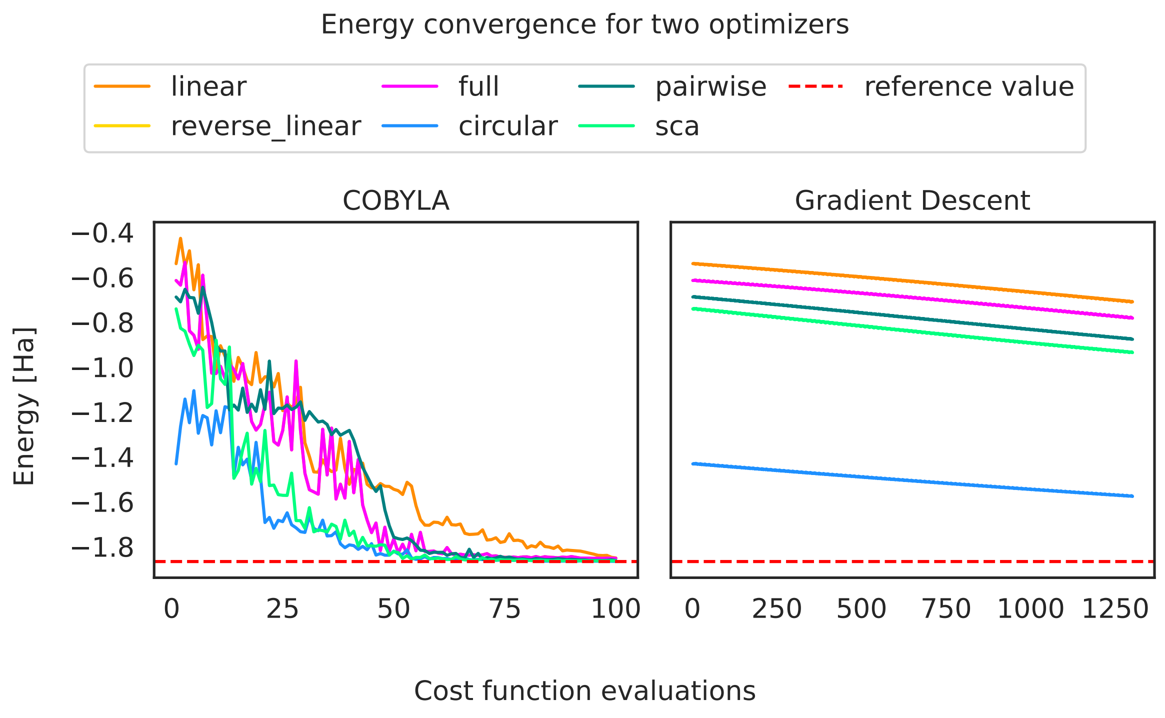


Figure 4.1: Examples of energy convergence for various optimizers

From the above figure, we can observe that not only convergence to the ground state energy but also cost function evaluations differ a lot. Some optimizers have a fixed number of cost function evaluations for each iteration, an amount of cost function evaluations in a single iteration depends on an optimizer's strategy. Another thing worth pointing out is that we benchmarked six ansatzes, however, we can see only five of them. The reverse linear ansatz is overlapped by the full ansatz since a matrix representation of the reverse linear ansatz is the same as the full ansatz. Observation of such big differences led us to the idea of benchmarking multiple ansatzes and optimizers and doing a further analysis of that. In the following subsections, we describe details of our benchmarking.

4.1.1 Ansatzes

In our benchmark, we used the same ansatzes as introduced in section 2.4.1, namely linear, reverse linear, full, circular, pairwise, and SCA. Additionally, from each ansatz, we created 3 variations, 1-layer, 2-layer, and 3-layer ansatzes. This yields a total of 18 parametrized quantum circuits that were tested with the VQE algorithm.

4.1.2 Hamiltonian

For all experiments, we used a 4-qubit Hamiltonian of a hydrogen molecule. We took it from a paper produced by Miháliková et al. [35] and it is defined as follows:

$$\begin{aligned}\hat{H}_{H_2}^{4-qubit} &= c_0 1 + c_1 Z_0 + c_2 Z_1 Z_0 + c_1 Z_2 + c_2 Z_3 Z_2 Z_1 + c_3 Z_1 + c_4 Z_2 Z_0 \\ &+ c_5 X_2 Z_1 X_0 + c_6 Z_3 X_2 X_0 + c_6 X_2 X_0 + c_5 Z_3 X_2 Z_1 X_0 \\ &+ c_7 Z_3 Z_2 Z_1 Z_0 + c_7 Z_2 Z_1 Z_0 + c_8 Z_3 Z_2 Z_0 + c_3 Z_3 Z_1,\end{aligned}$$

where coefficients are:

$$\begin{aligned} &c_0 = -0.80718, && c_1 = 0.17374, && c_2 = -0.23047, \\ &c_3 = 0.12149, && c_4 = 0.16940, && c_5 = -0.04509, \\ &c_6 = 0.04509, && c_7 = 0.16658, && c_8 = 0.17511. \end{aligned}$$

This Hamiltonian can be further simplified to 2 qubits, however, we retained this form to increase the complexity of the problem. The ground state energy of a hydrogen molecule is -1.8671050114542505, we calculated that using the *NumPyMinimumEigensolver* algorithm provided by Qiskit and we will use this value as a reference value for our experiments. Plotting energy convergence graphs for multiple optimizers and ansatzes does not show well the proximity of the resulting energy to the ground state energy. Additionally, interpreting such a vast amount of data is challenging, making it difficult to draw conclusions or present findings effectively on paper. Hence, we will consider the probability of reaching a chemical precision. Chemical precision refers to how closely individual measurements agree with the correct value [16]. The standard chemical precision is 0.0016 Ha, a familiar value among chemists.

4.1.3 Optimizers

After browsing some scientific articles, we did not find any definite answer to which optimizers work the best. This was also one of the reasons why we decided to test almost all optimizers that Qiskit offers. Furthermore, optimizers have a plethora of parameters and configuring them would be a nightmare, so we went with the default ones, except for the number of iterations. We set the number of iterations to 100 for each optimizer. Below are concise descriptions of all the tested optimizers, along with the summary table categorizing them by type.

AQGD (Analytical Quantum Gradient Descent) [36]:

- The algorithm proposed specifically for quantum problems.
- It tries to compute a gradient using a quantum circuit and also features variable step size.
- Despite its name including “gradient descent”, it remains gradient-free since the gradient is not computed in a standard analytical way.

NFT (Nakanishi-Fujii-Todo) [37]:

- The algorithm designed specifically for quantum-classical hybrid algorithms.
- It leverages the properties of quantum circuits and splits a problem into smaller subproblems by considering only a certain subset of parameters.

SPSA (Simultaneous Perturbation Stochastic Approximation) [38]:

- This algorithm approximates gradient hence, it cannot be considered as a true gradient-based algorithm.
- Each iteration requires only two cost function evaluations.

QNSPSA (Quantum Natural SPSA) [39]:

- This algorithm is tailored for quantum optimization problems, it builds upon standard SPSA algorithm and also leverages some properties of quantum circuits.

COBYLA (Constrained Optimization By Linear Approximation) [40]:

- Gradient-free and derivative-free method that examines a so-called “trusted region” of the current point and attempts to find the next point by linear approximation of a cost function.

Nelder Mead [41]:

- Derivative-free algorithm based on a simplex method which will evaluate $n + 1$ points that form simplex and gradually try to replace the worst point with a better one.

Powell [42]:

- Powell’s method does not require derivatives and ignores bounds and constraints.
- It iteratively examines orthogonal directions and performs one-dimensional minimization along each direction.

UMDA (Continuous Univariate Marginal Distribution Algorithm) [43]:

- The algorithm belongs to a family of evolutionary algorithms.
- It constructs a probabilistic model from the possible candidates and based on a previous iteration samples new candidates to reach better results.

Gradient Descent [44]:

- The standard gradient descent algorithm that moves by a specified step size in the direction of the steepest descent based on a calculated gradient.

CG (Conjugate Gradient) [45]:

- Unlike Gradient Descent which moves in the direction of the steepest descent, Conjugate Gradient picks a set of orthogonal directions and moves in each direction exactly once.

ADAM (Adaptive Moment Estimation) [46]:

- Gradient-based algorithm whose main feature is that it can adaptively adjust learning rates for each parameter during training, enabling efficient convergence.

AMSGRAD [47]:

- The variant of the ADAM algorithm that incorporates past gradients into the decision-making process, which also results in a higher memory consumption.

L_BFGS_B (Limited-memory BFGS Bound) [48]:

- BFGS algorithm is the quasi-Newton method.
- This memory-limited version approximates the original BFGS algorithm with a limited amount of memory and also enables us to define constraints for variables.

SLSQP (Sequential Least SQuares Programming) [49]:

- The quasi-Newton method that also incorporates techniques from quadratic programming.

TNC (Truncated Newton) [50]:

- It is also called Newton conjugate gradient because it uses the Conjugate gradient algorithm as an inner routine and also allows to set bounds for each variable.

Optimizer	**Type**
AQGD (Analytical Quantum Gradient Descent)	gradient-free
NFT (Nakanishi-Fujii-Todo)	gradient-free
QNSPSA (Quantum Natural SPSA)	gradient-free
SPSA (Simultaneous Perturbation Stochastic Approximation)	gradient-free
COBYLA (Constrained Optimization By Linear Approximation)	gradient-free
Nelder Mead	gradient-free
Powell	gradient-free
UMDA (Continuous Univariate Marginal Distribution Algorithm)	gradient-free
Gradient Descent	gradient-based
CG (Conjugate Gradient)	gradient-based
ADAM (Adaptive Moment Estimation)	gradient-based
AMSGRAD	gradient-based
L_BFGS_B (Limited-memory BFGS Bound)	gradient-based
SLSQP (Sequential Least SQuares Programming)	gradient-based
TNC (Truncated Newton)	gradient-based

Table 4.1: Categorized optimizers

4.1.4 Implementation and data exploration

In all the experiments we used the Qiskit [4] library, therefore all our code was written in Python programming language. We had to execute the VQE algorithm many times, and it imposed considerable time and computational requirements. To speed up our computations we created multiple processes and each process was responsible for a single optimizer. These processes were automatically distributed to available CPU cores.

All the data that we collected from the VQE runs we saved into a CSV file. This allowed us to later load the data into Pandas [51] data frame and perform data exploration. For data exploration, we primarily leveraged the Plotly [52] library, which has an easy-to-use API and creates nice interactive visualizations with few lines of code. However, all the visualizations included in this thesis were created using the Seaborn [53] library, which is built on top of Matplotlib [53].

It is important to mention that all computations we are considering here are ideal, meaning that we are not taking into account any noise. In total, we have 15 optimizers and 18 ansatzes, resulting in 270 possible combinations of the VQE execution. Each combination was executed 50 times with distinct initial points provided to an ansatz.

4.2 Ansatzes

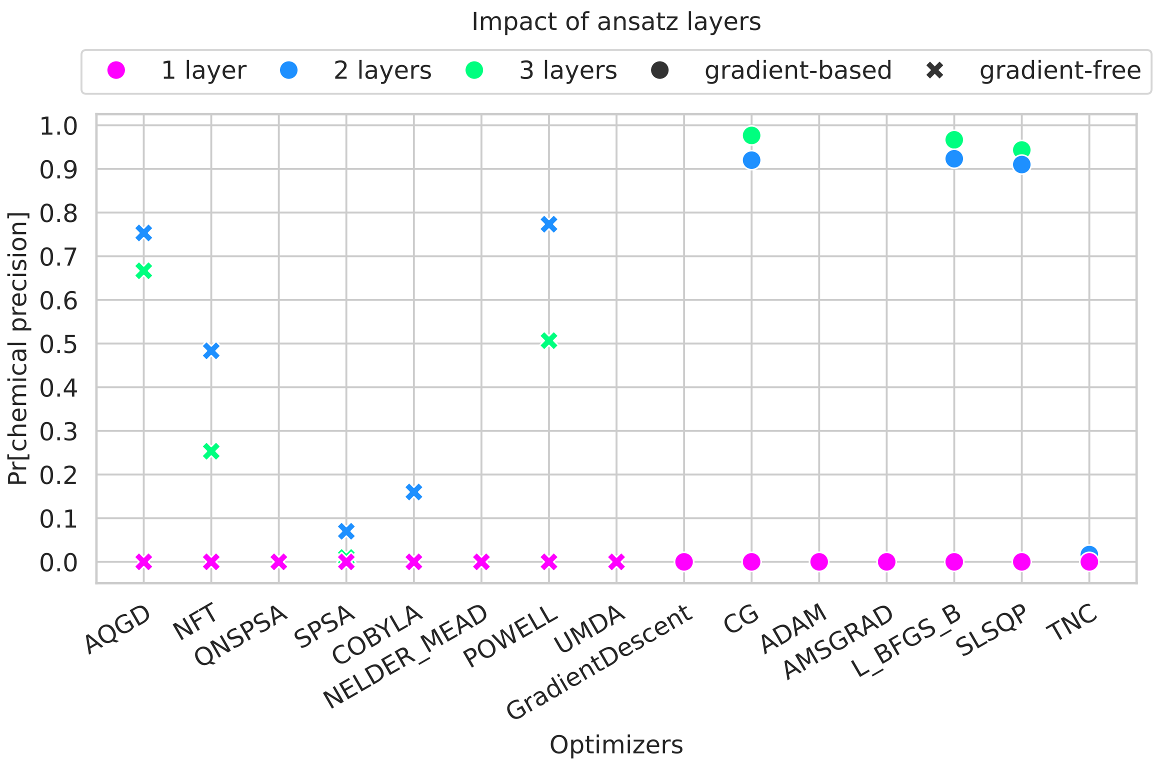


Figure 4.2: Achieving the chemical precision is unattainable using 1-layer ansatzes. Gradient-free optimizers reach better results with 2-layer ansatzes, while gradient-based optimizers perform better with 3-layer ansatzes.

An important finding we have made is the impossibility of achieving chemical precision using 1-layer ansatzes regardless of the optimizer. As depicted in Figure 4.2, the probability of achieving chemical precision remains consistently zero across all optimizers for 1-layer ansatzes. Consequently, in all subsequent analyses, we excluded 1-layer ansatzes and focused exclusively on 2-layer and 3-layer ansatzes. Additionally, we observed an intriguing trend that gradient-based optimizers tend to perform better with 3-layer ansatzes, whereas gradient-free optimizers achieve better results with 2-layer ansatzes. We attribute this finding to the number of parameters that need to be optimized. We guess that gradient-free algorithms can get lost more easily in bigger spaces and gradient-based optimizers can better navigate the space thanks to gradients. However, we have not taken further steps to verify this hypothesis.

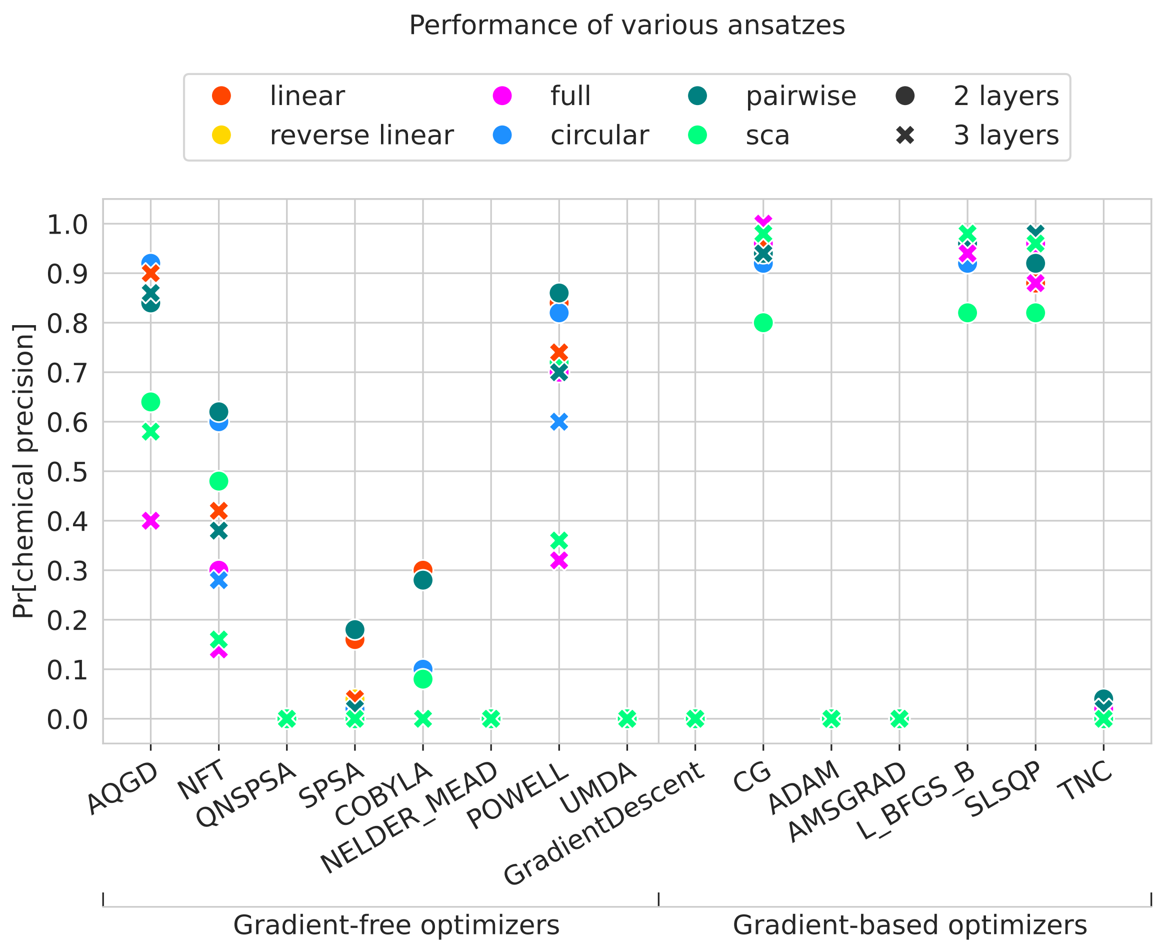


Figure 4.3: A more detailed version of the previous figure showing also types of ansatzes with a different number of layers. Gradient-based optimizers either achieve success across all tested ansatzes or completely fail. The choice of ansatz seems to have a greater impact with gradient-free optimizers.

In Figure 4.2, we were interested in ansatz layers. Figure 4.3 is very similar, but in addition to that, it also considers types of ansatzes. The figure may seem somewhat unclear because of the amount of overlapping data points. In general, it is not possible to say which ansatz is the best, but if we split our results into gradient-based and gradient-free optimizers we can observe some trends. When it comes to gradient-based optimizers, there are not any significant differences between the ansatzes. Either an optimization algorithm works and can reach a good result with any tested ansatz or does not work at all. On the other hand, the performance of the ansatzes with gradient-free optimizers is more fragmented. The SCA and full ansatzes seem to be the worst in combination with gradient-free optimizers. The best results are achieved with the pairwise, linear, and circular ansatzes. Surprisingly, there is a single combination of ansatz and optimizer that was able to reach a chemical precision in all 50 runs. The combination is constituted by the 3-layer full ansatz and *Conjugate gradient (CG)* optimizer.

4.3 Optimizers

The previous section indicated something about optimizers, but it was more geared towards the performance of ansatzes. In this section, we will try to discuss the performance of individual optimization algorithms in more detail by presenting the results in two distinct manners.

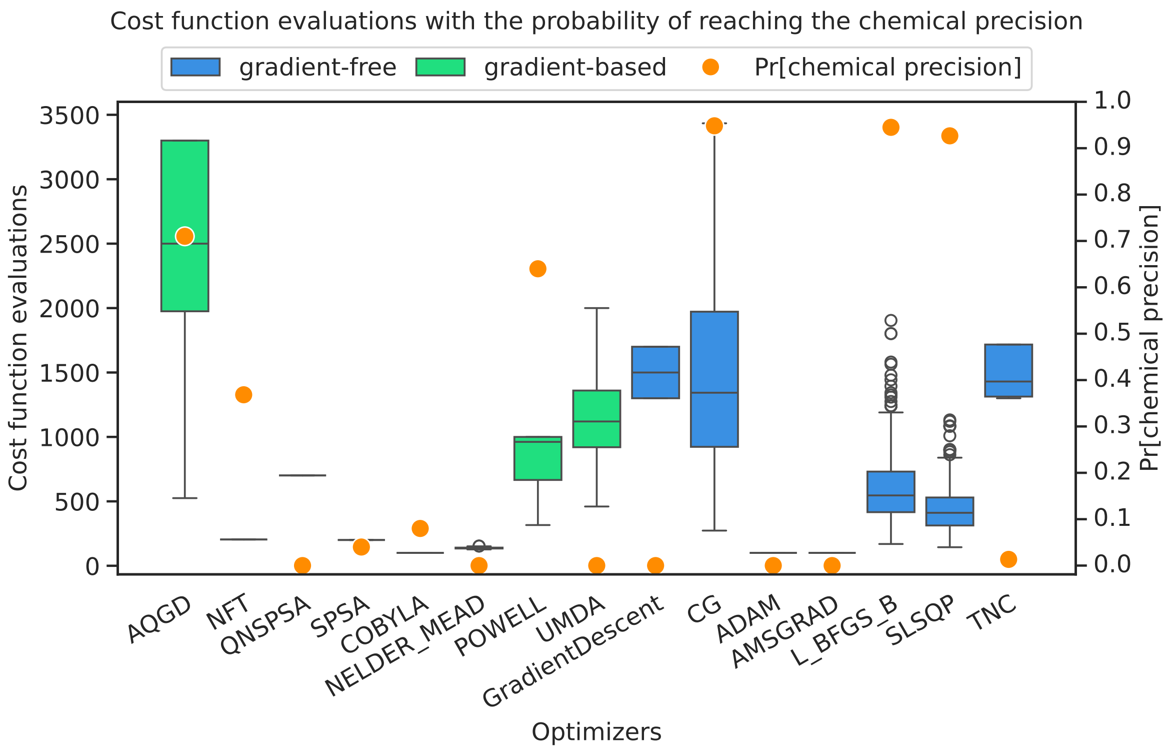


Figure 4.4: The relationship between the number of cost function evaluations and the probability of achieving chemical precision for various optimizers. The performance of some optimizers appears to be constrained by a fixed cost function evaluation count. However, there are some exceptions that achieved very good results, despite having fewer cost function evaluations.

Figure 4.4 depicts a visualization where the left y-axis shows the number of cost function evaluations, while the right y-axis shows the probability of reaching the chemical precision. The straight lines we can see on *NFT*, *QNSPSA*, *SPSA*, *COBYLA*, *ADAM*, and *AMSGRAD* mean that these optimizers have a fixed number of cost function evaluations. It seems that a fixed number of cost function evaluations can have a limiting impact on the optimizers. Multiple optimizers that have a smaller number of cost function evaluations are not very likely to reach the chemical precision, however, *L_BFGS_B* and *SLSQP* achieved very good results, despite the lower number of cost function evaluations. The *Conjugate gradient* has the best probability overall, however, it necessitates a considerable number of cost function evaluations. On the other hand, *Gradient descent* and *TNC* algorithms do not work at all even though the number of cost function evaluations is not so low.

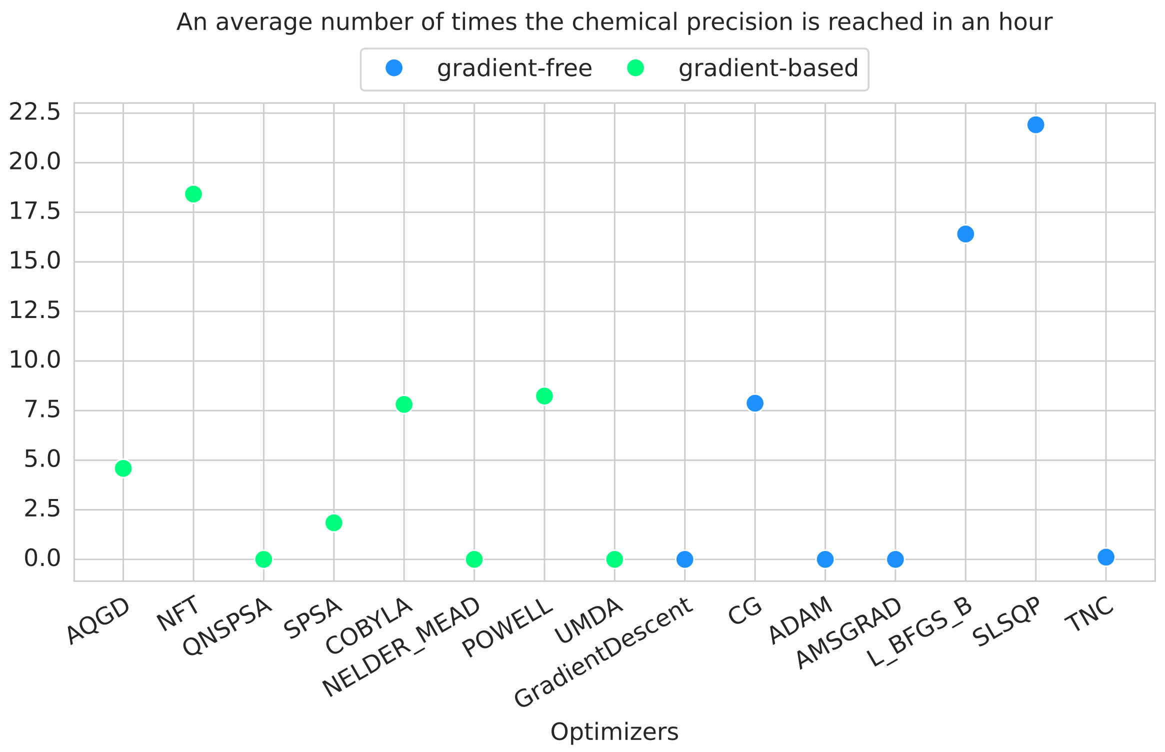


Figure 4.5: Another point of view on the data shown in the previous figure. This view favors fast optimizers and they can reach solid results despite the lower probability of reaching the chemical precision.

In Figure 4.5, we can see the average number of times the chemical precision is reached in an hour (on our hardware). This gives us a different view of the results. For instance, the *NFT* optimizer does not even have a 50% probability of reaching a chemical precision, however, due to its low amount of cost function evaluations, execution of the algorithm is fast and therefore we can reach the chemical precision more times in an hour than with other optimizers. On the other hand, the *AQGD* with a high number of cost function evaluations and with the probability of reaching chemical precision over 70% can yield correct results on average only nearly 5 times in an hour. This view of the data favors optimizers that are fast, they can achieve good results even though the probability of achieving chemical precision is low.

Initially, after running all the experiments, we did not consider the chemical precision and we thought that gradient-free optimizers perform better because of their faster energy convergence. However, after introducing the chemical precision, it turned out that the opposite is true. Gradient-free optimizers have difficulties getting close enough to the ground state energy.

As a clear winner seems the *SLSQP* algorithm that leads in both graphs. It has a very high probability of reaching the chemical precision, a smaller number of cost function evaluations, and it was able to reach the chemical precision with all tested ansatzes without any significant differences in results. Nevertheless, the best optimizer can vary from the situation. Sometimes, we strive to minimize the number of cost function evaluations since at the end of the day, it can be what we pay for. Around half of the optimizers have the probability of reaching the chemical precision of 0% or close to 0%, they were unable to tackle this problem. There may be a chance to enhance the results by increasing the number of iterations and fine-tuning the parameters of optimizers, particularly those utilizing gradients, as some of those gradient-based optimizers exhibit good performance. However, we have not taken any steps in that direction.

Conclusion

In this bachelor's thesis, we focused on a variational quantum eigensolver algorithm that was used to find the ground state energy of a hydrogen molecule. However, the performance of the VQE varies based on a chosen ansatz and optimization algorithm. We evaluated the performance of various ansatzes and optimization algorithms under ideal conditions on a 4-qubit representation of a hydrogen molecule.

The main outcome of our work is that gradient-based optimizers have a higher chance of yielding correct results than gradient-free optimizers. The gradient-based optimizers reached either very good results or completely failed regardless of the chosen ansatz. We assume that this notable difference in performance was primarily due to a fixed number of cost function evaluations. There could be a chance to enhance the performance of bad-performing optimizers, at least the gradient-based, by increasing the number of iterations and adjusting their parameters. On the other hand, gradient-free optimizers can operate faster, however, at a cost of a lower probability of reaching the chemical precision. Also, the choice of ansatz seems to have a greater impact than in gradient-based optimizers. Another intriguing finding is that gradient-free optimizers reach better results with 2-layer ansatzes and gradient-based with 3-layer ansatzes, 1-layer ansatzes do not work at all. The *SLSQP (Sequential Least SQuares Programming)* algorithm appears to be the best optimizer in terms of speed and probability of reaching the chemical precision. Overall, we think that a choice of optimizer is more important than a choice of ansatz, at least in ideal conditions where noise and errors are not present.

There are many ways how to advance our work. It would be interesting to try our benchmark with a broader set of problems or on a problem of a larger scale than just a hydrogen molecule and see whether our results can be transferred there. Moreover, we could hand-pick some optimizers, thoroughly investigate their characteristics, and adjust their parameters to maximize their effectiveness. Furthermore, experimenting with this benchmark on a simulator incorporating noise and errors would provide a more realistic assessment of the optimizers' capabilities. Another option is to execute this benchmark on a real quantum computer. However, this would be a costly approach.

Bibliography

[1] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J. Love, Alán Aspuru-Guzik, and Jeremy L. O'Brien. A variational eigenvalue solver on a photonic quantum processor. *Nature Communications*, 5(1), 2014.

[2] John Preskill. Quantum Computing in the NISQ era and beyond. *Quantum*, 2:79, 2018.

[3] Jules Tilly, Hongxiang Chen, Shuxiang Cao, Dario Picozzi, Kanav Setia, Ying Li, Edward Grant, Leonard Wossnig, Ivan Rungger, George H. Booth, and Jonathan Tennyson. The Variational Quantum Eigensolver: A review of methods and best practices. *Physics Reports*, 986, 2022.

[4] IBM Quantum Computing | Qiskit. `https://www.ibm.com/quantum/qiskit`. Retrieved April 20, 2024.

[5] George B. Arfken, Hans J. Weber, and Frank E. Harris. *Mathematical Methods for Physicists: A Comprehensive Guide*. Elsevier Science, 2013.

[6] Wikimedia Commons. Eigenvalue equation as a homothety on a vector. `https://upload.wikimedia.org/wikipedia/commons/5/58/Eigenvalue_equation.svg`, 2020. Retrieved April 12, 2024.

[7] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010.

[8] Wikimedia Commons. Bloch sphere, a geometrical representation of a two-level quantum system. `https://upload.wikimedia.org/wikipedia/commons/6/6b/Bloch_sphere.svg`, 2023. Retrieved February 13, 2024.

[9] Michał Stęchły. Variational Quantum Eigensolver explained. `https://www.mustythoughts.com/variational-quantum-eigensolver-explained`, 2019. Retrieved March 16, 2024.

[10] QuTiP - Quantum Toolbox in Python. `https://qutip.org/`. Retrieved May 22, 2024.

[11] Cirq | Google Quantum AI. `https://quantumai.google/cirq`. Retrieved April 20, 2024.

[12] PennyLane. `https://pennylane.ai/`. Retrieved April 20, 2024.

[13] Introduction to Q# & Quantum Development Kit - Azure Quantum | Microsoft Learn. `https://learn.microsoft.com/en-us/azure/quantum/overview-what-is-qsharp-and-qdk`. Retrieved April 20, 2024.

[14] Silq - What is Silq? `https://silq.ethz.ch/`. Retrieved April 20, 2024.

[15] Qiskit ecosystem. `https://qiskit.github.io/ecosystem/`. Retrieved March 3, 2024.

[16] Theodore L. Brown, Eugene H. LeMay, Bruce E. Bursten, Catherine J. Murphy, and Patrick M. Woodward. *Chemistry: The Central Science*. Pearson Prentice Hall, 12 edition, 2012.

[17] Hartree - Wikipedia. `https://en.wikipedia.org/wiki/Hartree`. Retrieved March 31, 2024.

[18] Light Website: Wavelength. `http://light.physics.auth.gr/enc/wavelength_en.html`. Retrieved April 6, 2024.

[19] Specify observables in the Pauli basis | IBM Quantum Documentation. `https://docs.quantum.ibm.com/build/specify-observables-pauli`. Retrieved March 25, 2024.

[20] Neil A. Gershenfeld. *The nature of mathematical modeling*. Cambridge University Press, 1999.

[21] Zoë Holmes, Kunal Sharma, Marco Cerezo, and Patrick J. Coles. Connecting ansatz expressibility to gradient magnitudes and barren plateaus, 2022.

[22] Nobuyuki Yoshioka, Hideaki Hakoshima, Yuichiro Matsuzaki, Yuuki Tokunaga, Yasunari Suzuki, and Suguru Endo. Generalized Quantum Subspace Expansion. *Physical Review Letters*, 129(2), 2022.

[23] Marco Cerezo, Akira Sone, Tyler Volkoff, Lukasz Cincio, and Patrick J. Coles. Cost function dependent barren plateaus in shallow parametrized quantum circuits. *Nature Communications*, 12(1), 2021.

[24] Lorenzo Leone, Salvatore F. E. Oliviero, Lukasz Cincio, and Marco Cerezo. On the practical usefulness of the Hardware Efficient Ansatz. 2022.

[25] Transpiler | IBM Quantum Documentation. `https://docs.quantum.ibm.com/api/qiskit/transpiler`. Retrieved March 03, 2024.

[26] Native gates and operations | IBM Quantum Documentation. `https://docs.quantum.ibm.com/run/native-gates`. Retrieved March 3, 2024.

[27] Alexander H. Jones and Zobia Khan. Reducing Error in Quantum Computing with Improved Circuit Design Methods. `https://pressbooks.howardcc.edu/jrip4/chapter/reducing-error-in-quantum-computing-with-improved-circuit-design-methods/`. Retrieved April 27, 2024.

[28] Matthew B. Hastings, Dave Wecker, Bela Bauer, and Matthias Troyer. Improving Quantum Algorithms for Quantum Chemistry, 2014.

[29] TwoLocal | IBM Quantum Documentation. `https://docs.quantum.ibm.com/api/qiskit/qiskit.circuit.library.TwoLocal`. Retrieved March 31, 2024.

[30] IonQ. What is Hybrid Quantum Computing? `https://ionq.com/resources/what-is-hybrid-quantum-computing`, 2024. Retrieved March 23, 2024.

[31] Portfolio Optimization - Qiskit Finance 0.4.1. `https://qiskit-community.github.io/qiskit-finance/tutorials/01_portfolio_optimization.html`. Retrieved April 28, 2024.

[32] Jos Thijssen. *The variational method for the Schrödinger equation*, pages 29–42. Cambridge University Press, 2007.

[33] Aurel Galántai. The theory of Newton's method. *Journal of Computational and Applied Mathematics*, 124(1):25–44, 2000. Numerical Analysis 2000. Vol. IV: Optimization and Nonlinear Equations.

[34] John E. Dennis and Jorge J. Moré. Quasi-Newton Methods, Motivation and Theory. *SIAM Review*, 19(1):46–89, 1977.

[35] Ivana Miháliková, Matej Pivoluska, Martin Plesch, Martin Friák, Daniel Nagaj, and Mojmír Šob. The Cost of Improving the Precision of the Variational Quantum Eigensolver for Quantum Chemistry. *Nanomaterials*, 12(2), 2022.

[36] AQGD - Qiskit Algorithms 0.3.0. `https://qiskit-community.github.io/qiskit-algorithms/stubs/qiskit_algorithms.optimizers.AQGD.html#qiskit_algorithms.optimizers.AQGD`. Retrieved April 23, 2024.

[37] Ken M. Nakanishi, Keisuke Fujii, and Synge Todo. Sequential minimal optimization for quantum-classical hybrid algorithms. *Physical Review Research*, 2(4), 2020.

[38] SPSA Algorithm. `https://www.jhuapl.edu/SPSA/`. Retrieved April 21, 2024.

[39] Julien Gacon, Christa Zoufal, Giuseppe Carleo, and Stefan Woerner. Simultaneous Perturbation Stochastic Approximation of the Quantum Fisher Information. *Quantum*, 2021.

[40] Gradient-free optimizers - OpenQAOA. `https://openqaoa.entropicalabs.com/optimizers/gradient-free-optimizers/#cobyla`. Retrieved April 21, 2024.

[41] John A. Nelder and Roger Mead. A Simplex Method for Function Minimization. *The Computer Journal*, 7(4):308–313, 1965.

[42] POWELL - Qiskit Algorithms 0.3.0. `https://qiskit-community.github.io/qiskit-algorithms/stubs/qiskit_algorithms.optimizers.POWELL.html#qiskit_algorithms.optimizers.POWELL`. Retrieved April 21, 2024.

[43] UMDA - Qiskit Algorithms 0.3.0. `https://qiskit-community.github.io/qiskit-algorithms/stubs/qiskit_algorithms.optimizers.UMDA.html`. Retrieved April 21, 2024.

[44] GradientDescent - Qiskit Algorithms 0.3.0. `https://qiskit-community.github.io/qiskit-algorithms/stubs/qiskit_algorithms.optimizers.GradientDescent.html#qiskit_algorithms.optimizers.GradientDescent`. Retrieved April 21, 2024.

[45] Jonathan R. Shewchuk. An Introduction to the Conjugate Gradient Method Without the Agonizing Pain, 1994.

[46] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization, 2017.

[47] Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. On the Convergence of Adam and Beyond, 2019.

[48] L_BFGS_B - Qiskit Algorithms 0.3.0. `https://qiskit-community.github.io/qiskit-algorithms/stubs/qiskit_algorithms.optimizers.L_BFGS_B.html#qiskit_algorithms.optimizers.L_BFGS_B`. Retrieved April 20, 2024.

[49] Dieter Kraft. A software package for sequential quadratic programming. *The Computer Journal*, 1988.

[50] TNC - Qiskit Algorithms 0.3.0. `https://qiskit-community.github.io/qiskit-algorithms/stubs/qiskit_algorithms.optimizers.TNC.html#qiskit_algorithms.optimizers.TNC`. Retrieved April 20, 2024.

[51] Pandas - Python Data Analysis Library. `https://pandas.pydata.org/`. Retrieved May 22, 2024.

[52] Plotly Python Graphing Library. `https://plotly.com/python/`. Retrieved on May 2, 2024.

[53] Seaborn: statistical data visualization. `https://seaborn.pydata.org/`. Retrieved on May 2, 2024.

Appendix A

Source code

All source files for this project are publicly available in this GitHub repository: `https://github.com/misosvec/optimization-of-variational-quantum-eigensolvers`. Files `ansatzes.py` and `optimizers.py` handle an initialization of ansatzes and optimizers. File `benchmark.py` runs the VQE in a multiprocessing manner and saves produced data to a CSV file. Juptyer notebook `visualizations.ipynb` contains all visualizations of circuits, qubits, and gates used in our thesis. Plots representing results are located in file `results.ipynb`.