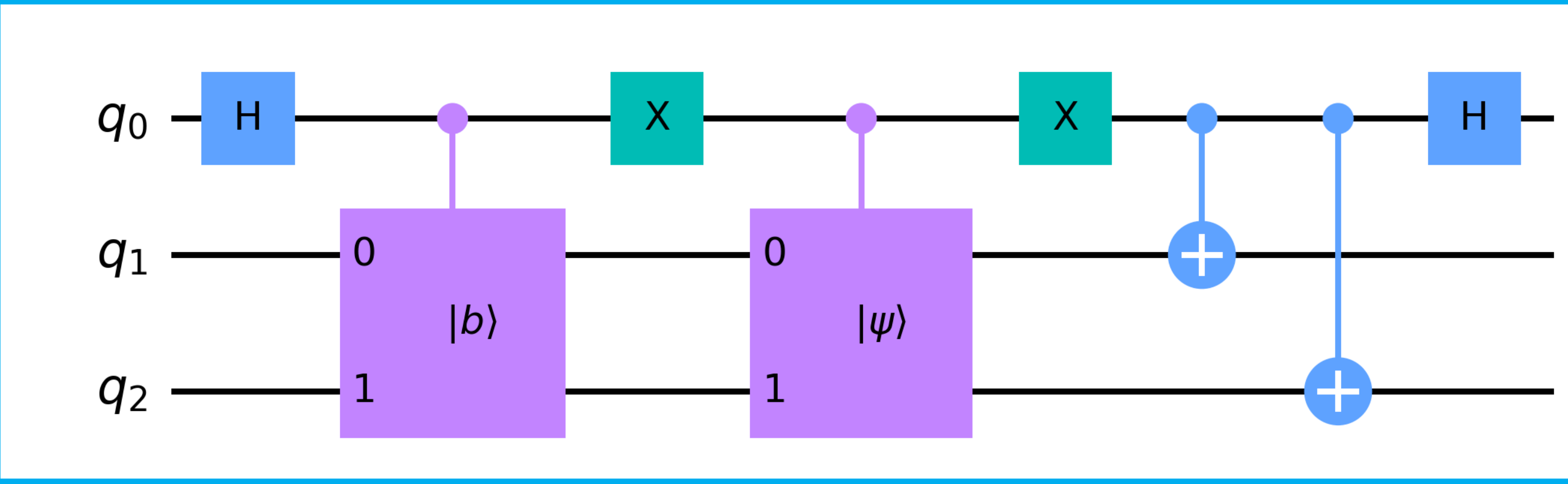
VQLS Read the Fine Print: Practical Challenges for Solving the Poisson Equation by Means of a Variational Quantum Linear Solver

T. R. Verduyn



VQLS Read the Fine Print: Practical Challenges for Solving the Poisson Equation by Means of a Variational Quantum Linear Solver

by

T. R. Verduyn

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Monday December 5, 2022 at 2:00 PM.

Student number:	4552695	
Project duration:	February 1, 2022 – December 5, 2022	
Thesis committee:	Dr. ir. M.I. Gerritsma,	TU Delft, supervisor

TU Delft

Abstract

The need of computational power for engineering applications has been ever increasing and with classical computers approaching their physical limits, new ways of improvement have to be investigated. One of the promising solutions is quantum computing. Most engineering problems require solving a system of linear equations of higher dimensions and the quantum algorithm known as the HHL algorithm, provides exponential speedup over classical methods. Near-term Noisy-Intermediate Scale Quantum (NISQ) computers are far from the technological level to implement this algorithm. Thus a different algorithm suitable for NISQ computers, named Variational Quantum Linear Solver (VQLS), is devised. Variational algorithms are heuristic in nature, since they use an ansatz which prepares a quantum state based on classical input parameters. VQLS is a hybrid algorithm, where the cost function is evaluated on a quantum computer and a classical optimizer is used to optimize the ansatz parameters. The minimum of this cost function corresponds to the solution of the problem.

In this work the Poisson equation is discretized, such that a linear system in the form of $Ax = b$ is obtained. This system is then solved by means of VQLS. An important part in this process is finding an efficient decomposition of the A-matrix into a linear combinations of Hermitian operators. A decomposition into Pauli gates has an exponentially increasing number of terms in the number of qubits [14]. An efficient decomposition by using raising and lowering operators is introduced in [44], however this work did not mention anything about the trainability of the ansatz and occurrence of barren plateaus. Barren plateaus are areas in the cost function landscape where the variance of the cost function gradient vanishes exponentially in the number of qubits. Meaning that if such a plateau does occur, the minimization of the cost function becomes very difficult as the problem size is increased.

The aim of this work is to see what the practical limitations are when solving the Poisson equation by means of VQLS. It is expected that the main limitation will stem from the fact whether the efficient decomposition suffers from barren plateaus. This is investigated by using a simulated quantum implementation of the VQLS algorithm, but also by using a simplified method based on retrieving the quantum state from the ansatz and then applying linear algebra in order to evaluate the cost function. This second approach obtained identical results to the quantum implementation and was several orders of magnitude faster in run times. Qiskit is used to simulate the quantum machines.

The main results obtained during the study show that, while solving the Poisson equation with an efficient decomposition into raising and lowering operators, barren plateaus indeed do occur. This means that smaller problems with 2 and 3 qubit are still solvable, however, as the number of qubits is increased the algorithm becomes unfeasible to use due to the exponential increase of the number of shots needed to obtain a sufficient accuracy. This is detrimental to the trainability of the ansatz and makes solving problems of meaningful size effectively impossible. These barren plateaus are caused by the scaling of the individual cost function terms as the problem size increases. Primarily this is a result of the fact that quantum states are normalized vectors and can only interact with Hermitian operators. This problem seems inherent when solving the Poisson equation in combination with VQLS and thus is a very challenging task to solve.

Contents

List of Figures

1 Introduction

During the last few decades, the available computational power has drastically increased. Where computers used to be working with kilobytes and megabytes of data only a few decades ago, currently data processing in the order of terabytes or even exabytes has become common. The problems that engineers are trying to solve also grow in size along with this increase. Modern state-of-the-art computational fluid dynamics (CFD) simulations may require up to billions of cells, meaning that immense amount of computations are being processed. In 1965 Gordon Moore observed that the number of transistors in a circuit roughly doubled every year and he predicted that this trend would continue in the future. So far his law has shown to be a very good prediction. However, Shalf et al. [72] have shown that, as technology keeps advancing, the physical limits of chips will be approached and at some point in the near future Moore's Law will not hold anymore. This means that other ways of scaling computational capacity have to be investigated. One of the promising looking solutions is quantum computing. Instead of standard bits, which represent zeros and ones, a quantum computer uses qubits which can represent a zero, a one or a superposition of both. Next to the principle of superposition, quantum computing also requires entanglement, which means that states of different qubits are linked together and that a single qubit cannot be described independently from the others. By using both superposition and entanglement, quantum algorithms can achieve up to exponential speedup in scaling of the problem size compared to classical methods and greatly reduce the computational effort required to solve large problems.

In 2009 Harrow, Hassidim and Lloyd [30] devised a quantum algorithm that is capable of solving sparse linear systems with exponential speedup over classical solvers. Although the HHL algorithm is very promising in theory, it is not feasible for near-term implementation on existing quantum hardware. Currently quantum computers are at the so-called Noisy Intermediate-Scale Quantum (NISQ) stage [65]. This is the era where quantum computers contain between fifty to a hundred qubits and still are suffering from noise effects. This makes the implementation of an algorithm such as HHL infeasible for the near future. In the paper named *Read the Fine Print* by Scott Aaronson [1], a list of caveats for the implementation of the HHL algorithm are layed out. At first sight the HHL algorithm seems a very promising algorithm. However when one wants to make use of it, there are some major caveats hindering an efficient implementation. Aaronson showed that even though the algorithm itself theoretically allows for exponential speedup over classical methods, in practice this might be much more difficult to achieve.

Currently the hopes for quantum advantage are set on variational quantum algorithms (VQA). These types of algorithms only require a shallow quantum circuit. Since the circuits used for these algorithms contain fewer gates they minimize the amount of noise induced by the NISQ devices. The VQA alternative closest to the HHL algorithm is called the variational quantum linear solver (VQLS). This algorithm is conceived by Bravo-Prieto et al. [10] and solves a linear system in the form of $A|x\rangle \propto |b\rangle$. Variational algorithms are hybrid algorithms as they consist of both a classical and a quantum algorithm. Their objective is to use a classical optimizer to minimize a cost function, which is evaluated on a quantum computer. An ansatz is used to prepare a quantum state such that $|\psi(\theta)\rangle = V(\theta)|0\rangle$, where $|\psi(\theta_{opt})\rangle$

equals the solution vector $|x\rangle$. $|0\rangle$ is the initial zero state and $V(\theta)$ the ansatz operator based on the input parameters θ. The input parameters for the ansatz are then optimized classically such that the cost function is minimized. The ansatz plays an important role in VQAs and is the starting point of the algorithm. In the present day a lot of research is being performed in the area of VQAs and the avoidance of barren plateaus. Barren plateaus are areas in the cost function landscape where the gradient vanishes exponentially in the number of qubits. With the presence of barren plateaus, a quantum circuit has to be sampled an exponential number of times in order to obtained the optimized solution [50].

This thesis work aims at gaining a better understanding of why solving the Poisson equation by means of VQLS, is so difficult. The Poisson equation is chosen as a test case since it is relevant for a wide variety of engineering and physics problems and plays a role in, for example, the incompressible Navier-Stokes equations used in most CFD applications. Similarly to the work of Scott Aaronson this work tries to find what the caveats are for an efficient implementation of the VQLS algorithm when solving the Poisson equation. On a final note, it is assumed that the reader has some basic knowledge on quantum mechanics and for example knows what a Hilbert space or Bloch sphere is. The more specific aspects of quantum computing and quantum circuits will be explained in this work.

1.1. Research Questions and Objectives

The goal and objective of this thesis is to see what the limitations and hurdles are for solving the one-dimensional Poisson equation by means of a Variational Quantum Linear Solver (VQLS). The Poisson equation is taken as a test case, as it is an elliptic differential equation and non-trivial to solve by VQLS. Often times in literature the VQLS algorithm is used to solve very simple problems, which have limited use cases in the real world. By showing whether the VQLS algorithm can solve a more difficult problem, conclusions can be made on its usefulness in the future. In previous work [14], it has been shown that solving this equation by means of VQLS is quite challenging. In that work a decomposition of the discretized Poisson equation by Pauli operators was used. By following a slightly different approach in this thesis, based on an efficient decomposition in raising and lowering operators as introduced by [44], it is hoped that new insights can lead to a better working of the algorithm.

The thesis research objective is as follows:

> Investigate whether the Variational Quantum Linear Solver is a suitable method for solving the Poisson equation, by looking at what the practical challenges of its implementation are.

This objective can be reached by answering the main research question of this work:

> What are the practical challenges when solving the Poisson equation by means of the Variational Quantum Linear Solver?

In order to answer this question, several sub-questions are setup:

1. Does the VQLS algorithm suffer from barren plateaus when an efficient decomposition consisting of raising and lowering operators is used?
2. What causes vanishing of the gradients for certain cost functions?
3. Can changes to the cost functions help improve the trainability of the ansatz?
4. How can information of the quantum state be efficiently extracted and used classically?

By answering these questions, a structured approach to achieving the main research objective is used.

1.2. Thesis Outline

The structure of this thesis report is as follows. First some background information on the topic of quantum computing is given in Chapter 2. Here most of the information for understanding of further chapters and sections of the report is given. In Chapter 3 further details of the implementation of the VQLS algorithm are given. In Chapter 4, the methodology which is used to obtain the results is explained. The results from running the VQLS algorithm and other tests are elaborated Chapter 5. Finally in Chapter 6, the conclusions drawn from the results are discussed and recommendations for future works are layed out.

2 Literature Review

This section covers information important for understanding later parts of the thesis. First a short introduction to quantum computing is given and the types of quantum computers is explained. After that, the current state of quantum computing is briefly covered, followed by explanation of the Hadamard test and swap test. Then the HHL algorithm is explained since it is the original method of solving a linear system on a quantum computer. Then the variational quantum linear solver (VQLS) is introduced. Since VQLS is the algorithm that is used to solve the Poisson equation during this thesis, all of its components are explained in more details. These are: the Ansatz, Cost functions, Barren plateaus, the decomposition of the A-matrix and the classical optimizer.

2.1. General Introduction to Quantum Computing

Research regarding quantum computing has gained a lot of attention over the recent decades. After Manin proposed the idea of quantum computing in 1980 [48] and Feynman in 1982 [25], the hopes for constructing a physical quantum computer raised and the first realization of an algorithm on a quantum computer was made in 1998 [35]. In the mean time gate-based quantum computers have increased from 2-qubits to over 100 qubits in size, while adiabatic quantum computers already contain over a 1000 qubits. The goal of quantum computers is to obtain speedup over classical methods for all kinds of problems. These can range from solving linear systems to improving search algorithms or storing data. By using faster algorithms, quantum computers might also help with reducing energy usage of supercomputers [12]. Current supercomputers have an immense amount of energy usage and by improving the quality of quantum computers and the qubits within them, the processing power of such computers can increase without a proportional increase in energy costs[1].

2.1.1. Differences between Classical and Quantum Computers

Before one can understand how a quantum computer works, it is important to understand some of the basics of classical computation. In short a classical computer works with bits, that either represent a zero or a one. By manipulating these bits with gates, all sorts of computations can be performed, which ultimately lead to applications such as the internet or CFD software. A quantum computer uses qubits, which represent either a zero, a one, or a superposition of both at the same time. A superposition of a single qubit can be written as $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, where α and β are normalized coefficients such that $|\alpha|^2 + |\beta|^2 = 1$. In combination with a quantum phenomenon called entanglement, quantum algorithms can be constructed. Quantum entanglement is the phenomenon where two qubits are correlated beyond what is classically possible and cannot be completely described independently from each other. Entanglement and superposition are both fundamentally different from what is classically possible and gives access to a new way of computations. By combining these two effects and using them in sophisticated ways, new algorithms can be constructed that provide exponential speedup over classical methods. Often times quantum computing is viewed as something radically different from classical computing, however a better way to state it, is that classical computing is a limited form of

[1]https://qz.com/1566061/quantum-computing-will-change-the-way-the-world-uses-energy/, https://www.azoquantum.com/Article.aspx?ArticleID=136

quantum computing, where the effects of entanglement and superposition are not usable [8]. However, given sufficient time any problem solvable on a quantum computer can also be solved on a classical computer, based on the Church-Turing thesis.

In order to build a quantum computer DiVincenzo introduced 5 criteria: [20]:

1. A scalable physical system with well characterized qubits
2. The ability to initialize the state of the qubits to a simple fiducial state, such as $|000\rangle$
3. Long relevant decoherence times, much longer than the gate operation time
4. A ”universal” set of quantum gates
5. A qubit-specific measurement capability

When these 5 criteria are met, a fully functioning quantum computer is constructed. If one of the points is missing in a quantum device it will not be fully functional or scalable. Currently quantum computers are still challenged on multiple of these criteria. For example, scaling of the number of qubits is difficult, as the number of direct connections between each qubit drastically increases. As a result qubits are only partially connected in most current devices. The limited decoherence times also does not allow for running circuits of any arbitrary length. Finally, implementation of quantum gates, measurements and readout of qubits also has limited accuracy.

2.1.2. Types of Quantum Computers

Quantum computers can be divided into two different categories. Quantum annealers (QA) and gate-based quantum computers (GBQC). Quantum annealers borrow their name from a classical process used in material sciences, called thermal annealing. Here a material is heated, and then slowly cooled such that its properties change. QAs work with a completely different principle as GBQC. GBQC are closer to classical computers in the sense that they are also operated by gates. The following sections will explained the differences between the two types in more detail.

2.1.3. Quantum Annealer

Quantum annealers are useful for optimization problems. Annealing is a process where a global minimum is found over a set of candidate solutions [22]. A simplified representation is shown in fig. 2.1. First the qubits are brought into a superposition and are in the lowest-energy eigenstate of the initial Hamiltonian, after which the the Hamiltonian is slowly morphed into the problem Hamiltonian. This morphing of the problem changes the minimum energy levels, and when it is performed slow enough a global minimum will be obtained [34]. This process is closely related to adiabatic evolution as described by Fahri et al. [24]. The explanation given here is very simplified and for further information [22, 24, 34] can be consulted.

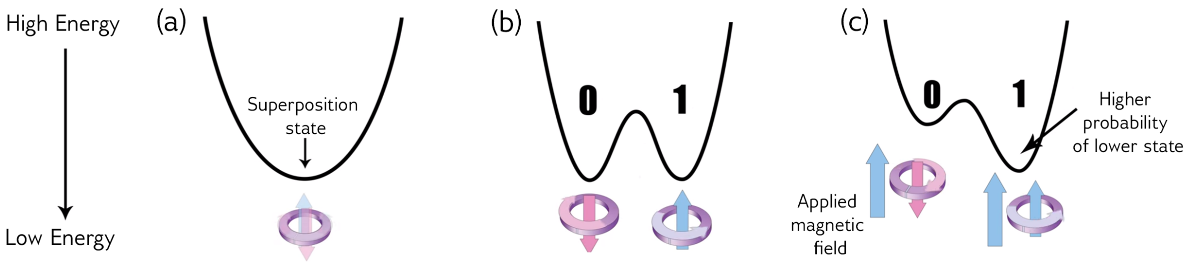


Figure 2.1: Energy diagram of the quantum annealing process. [34]

Before a problem can be solved on a QA, it has to be formed into either a Quadratic Unconstrained Binary Optimization (QUBO) problem or an Ising format. The QUBO and Ising formats are NP-hard to solve and can be adapted to a variety of problems [46]. So obtaining a solution to these problems classically is a non-trivial task. Study by Vinci et al. [79] has shown that QA induces non-stoquastic interactions (which is a quantum form of a classical non-stochastic matrix, where all off-diagonal elements in the Hamiltonian are real and non-positive), this effect prohibits classically efficient simulation of QA. The output of a quantum annealer is a bit string, which corresponds to the lowest-energy state of the problem Hamiltonian. By sampling the solution multiple times, a higher confidence of the accuracy of the solution can be obtained.

The most well known quantum annealers are from a company called D-Wave. Within the scientific community D-Wave has sparked a lot of controversy. Currently their largest system contains over 2000 qubits [2], and is significantly larger than that of the largest GBQCs, which are in the order of 50-100 qubits. However, the D-Waves system is not fully connected, meaning that the effective number of qubits is smaller. The degree of ”Quantumness” that actually happens on a D-wave machine has been questioned [76]. It is not fully clear what happens on the machine and it is said to operate like a black-box. The D-wave machines fall under a type called the noisy adiabatic quantum computer. Solving a structural optimization problem in the form of QUBO on a D-wave machine has been performed by K. Wills and was successful for small problems, however, scaling them to larger problems proved to be difficult [87].

Quantum Annealers are polynomially equivalent to the standard circuit-based quantum computers [3]. This means that standard quantum computers can be efficiently simulated on a quantum annealer. However, whether this is also the case in practice is not clear, as has been shown with the controversy regarding the level of quantum computing in D-Wave machines. Another study has shown that fundamental limitations in the temperature scaling for QAs limits them as competitive scalable optimizers [4]. However, despite all of its controversy D-Wave has shown to obtain results in a variety of problems [19, 56]. And small linear systems of equations have been solved by means of QUBO on a D-wave QA [36], however the success rate of finding the minimum energy quickly drops off with 63.16% for a two-dimensional case and 1.38% for a three-dimensional case. Other studies have also shown that the scalability of quantum annealers is uncertain. Such as Rønnow et al. [67] stating that no evidence for quantum speedup in a random spin glass instance problem could be found on a 503 qubit quantum annealer. However, they also state that quantum speedup cannot be ruled out for different problem classes. Mandrà et al. [47] showed that the glass-spin benchmarks that were solvable on a quantum annealer are also solvable with polynomial time on classical devices, and thus do not specifically indicate anything of a quantum speedup. The problem regarding scalability of quantum annealers is also addressed in the works of [65], and here it is mentioned that there is no theoretical argument or experimental evidence indicating that quantum annealers can help achieving speedup over classical methods. Since the scaling of quantum annealers to larger problems and their exact working is still in question, they are not considered for this thesis work.

2.1.4. Gate-Based Quantum Computer

Gate-based quantum computers are operated, just like classical computers, with gates. By constructing a quantum circuit consisting of qubits and gates operating on them, algorithms can be formed. One difference between classical and quantum gates is that quantum gates must be reversible [57]. This means that, based on the output state, it must be possible to recreate the initial state, and that thus no information is lost within the process. Mathematically this means that a quantum circuit (without measurements) represents a unitary matrix. Other operations such as measurements (observables) are not tied to this constraint. The requirement for a quantum gate to be unitary is the only constraint given for a quantum gate and is a result from the time evolution postulate. A matrix (or gate) is unitary when $U^\dagger U = I$, where $U^\dagger$ is the complex conjugate of matrix U [57]. Since these gates are unitary, they are also reversible. The gates acting on a GBQC are either single qubit or multi qubit gates and correspond to rotations around the Bloch sphere. The most well known gates are the Pauli X, Y and Z gates, their corresponding unitaries are shown below:

$$\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad \sigma_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad \sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{2.1}$$

The set of Pauli gates and the identity gate forms a basis set for all Hermitian matrices of size 2^n by 2^n, meaning that any Hermitian matrix of this size can be decomposed into the Pauli gates. Another important group of gates for constructing quantum circuits are the R_x, R_y and R_z gates, as shown

[2]https://www.dwavesys.com/solutions-and-products/systems/

in eq. (2.2). These gates are cyclic and require a certain angle as input. For the single qubit case, initialized with the angle $\theta = \pi$, the rotation gates correspond to the Pauli gate for their respective axis up to a global phase.

$$R_x(\theta) = \begin{pmatrix} cos(\frac{\theta}{2}) & -isin(\frac{\theta}{2}) \\ -isin(\frac{\theta}{2}) & cos(\frac{\theta}{2}) \end{pmatrix}, \quad R_y(\theta) = \begin{pmatrix} cos(\frac{\theta}{2}) & -sin(\frac{\theta}{2}) \\ sin(\frac{\theta}{2}) & cos(\frac{\theta}{2}) \end{pmatrix}, \quad R_z(\theta) = \begin{pmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{pmatrix} \tag{2.2}$$

Another important gate is the Hadamard gate, this gate is described by eq. (2.3). This gate is used to bring a qubit into an equal superposition of $|0\rangle$ and $|1\rangle$ by rotating away from the Bloch sphere poles.

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{2.3}$$

Another single qubit gate relevant to this thesis, is the phase gate, or S-gate. This gate rotates the quantum state 90 degrees around the z-axis. As can be seen in the first matrix in eq. (2.4), it does not affect the $|0\rangle$ state. It does however have an effect on the $|1\rangle$ state, as $S|1\rangle = i|1\rangle$. This way it allows complex amplitudes to be generated. The conjugate transpose of this gate is defined as $S^\dagger$ and is shown on the right in eq. (2.4).

$$S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}, \quad S^\dagger = \begin{pmatrix} 1 & 0 \\ 0 & -i \end{pmatrix} \tag{2.4}$$

For multi qubit gates, the most used ones are the CNOT, SWAP and CU (Controlled-U) gates and their simplest versions are shown in eq. (2.5). The CNOT gate flips the second qubit based on the value of the first qubit and is used to introduce entanglement. The SWAP gate, as the name implies, swaps the two qubits it is connected to. The CU gate implements a single or multi-qubit gate based on the value of the control qubit. In eq. (2.5) the case of a controlled single qubit operations is shown, where $u_{i,j}$ indicate the entries of the single qubit matrix. The gates mentioned here are only a limited part of all gates available and for an overview of more gates other sources such as the Qiskit textbook[3] or the book by Nielsen and Chuang [57] can be used.

$$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \quad SWAP = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad CU = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & u_{00} & u_{01} \\ 0 & 0 & u_{10} & u_{11} \end{pmatrix} \tag{2.5}$$

Some examples of well known quantum algorithms are: Shor's algorithm for prime factoring of integers, Grover's algorithm for quantum search, or the HHL algorithm used for solving linear systems of equations [4]. For an overview of quantum algorithms, the overview by [52] can be consulted. Since this overview stems from 2016, it does not contain any information regarding variational algorithms.

2.1.5. Quantum Computations

How exactly are these quantum computers used to solve problems? There is a large variety of methods and algorithms used for obtaining solutions of all sorts of problems. In essence there are two different quantum algorithms, some algorithms only have to be run once (assuming perfect quantum hardware), while others, such as variational algorithms require to be executed multiple times. For example, determining whether a Boolean function is balanced or constant in the Deutsch-Josza algorithm or what the hidden bit string in the Bernsteing-Vazirani algorithm is, only requires the algorithm to be executed a single time. For other algorithms, and especially variational algorithms, the circuits have to be run multiple times. In these cases a certain cost function is evaluated, based on the quantum state initialized at the start of the algorithm. By defining the cost function such that the minimum corresponds to the solution, the solution can be found by varying the input and finding the minimal cost. These cost

[3]`https://qiskit.org/textbook`

[4]For an overview of more algorithms `https://quantumalgorithmzoo.org/` can be consulted. This website contains over 400 cited research papers on quantum algorithms.

functions often rely on the evaluation of an expectation value and are a very important part in GBQC. The expectation value gives, as the name implies, the expected value when measuring an operator with respect to a quantum state. For an arbitrary quantum state $|\psi\rangle$ and an observable A, measuring the expectation value is done as show in eq. (2.6). An interesting point is that the observables themselves are not limited to being Hermitian [32]. Hermitian observables are merely a subclass of all measurable observables, which are the normal operators. A requirement is that the eigenstates of an observable are measurable. The observable A will correspond to a measurable operator, meaning that it corresponds to doing a matrix vector product. So by means of standard linear algebra it is easy to compute for smaller systems in a classical way.

$$\langle A \rangle = \langle \psi | A | \psi \rangle \tag{2.6}$$

Performing this computation on a quantum computer requires a bit more processing of the data. As running a quantum circuit will only yield individual (qu)bit values, a circuit has to be run multiple times to render an approximation to the solution. Circuits that can be used for this type of measuring of the expectation values are explained in section 2.3.

An example of a variational algorithm used on a GBQC is the variation quantum linear solver (VQLS). For the VQLS algorithm, which solves a linear system in the form of $Ax = b$, the vectors x and b are represented as quantum states. These vectors both have to be normalized, as quantum states are represented by normalized vectors. By using a parameterized ansatz, which is then optimized, finally a quantum state $|x\rangle$ is obtained, which is proportional to the classical solution vector x. This type of data structure is called amplitude encoding, since the solution information is encoded on the amplitudes of the quantum state.

2.1.6. Universality

An important aspect of GBQC is universality, which is the process of converting any set of arbitrary input, into any set of outputs. For classical computation universality is often shown by means of the universal Turing machine. For quantum computers the same has been proven [45], and an infinite list of quantum gate sets that are universal can be found. Simple examples of universal gate sets are R_x, R_y, R_z and CNOT, or CNOT and any single-qubit real gate that does not preserve the computational basis, and is not the Hadamard-gate [75]. From Knill's theorem, which states that a GBQC that contains only Clifford group gates and Pauli group measurements can be efficiently simulated classically [26], it can be concluded that quantum computing is indeed more powerful than classical computation. Since the set of Clifford gates (CNOT, S, H) and Pauli gates are not universal.

2.2. Current State of Quantum Computing

Currently quantum computing finds itself in the so-called Noisy Intermediate-Scale Quantum (NISQ) era, as devised by [65]. The NISQ era refers to the time period where quantum computers are between 50-100 qubits in size, still suffer from noise effects and do not yet make use of quantum error correction. With the current state of quantum hardware, simulation on classical devices can still keep up. In a study conducted by Zhou et al. [89], the performance of simulated quantum computers on classical devices was studied. Here the main observation is that fidelities of 99% are cheaper to achieve on a classical computer than on a quantum computer. Here a 2D-grid of qubits is simulated. By not simulating the complete Hilbert space, the required memory is reduced and simulation of larger systems is viable. The implication of this study is still questionable. A fidelity of 99% can be either a very good or mediocre result, depending on the area and type of research. For the current state of quantum computing, a 99% fidelity is indeed on the higher end of the spectrum, but this might change in the coming decades as hardware improves. As quantum computers increase in size, simulating them will become exponentially more difficult as the state vector size doubles with each additional qubit.

An important step for quantum computation will be to prove its supremacy over classical computers. Quantum supremacy means that a problem is solved on a quantum computer which would take a classical computer an immense amount of time. In 2019, a group of scientists from Google [7], claimed to have shown quantum supremacy for a problem, which is claimed to take classical computers over 10,000 years to compute. However, in 2021 [59], showed that they could compute the same results within a reasonable amount of time, meaning that quantum supremacy still has not been proven conclusively.

2.3. Standard Forms of Quantum Circuits

Moving on from a brief introduction of the topic of quantum computing, now some quantum circuits used to evaluate expectation values are introduced. These are the Hadamard test and Swap test. The Hadamard test is used to compute the expectation value between a quantum state and a unitary operation, while the swap test is used to measure the fidelity (overlap) between two quantum states. Both algorithms are important for VQAs and are explained in more detail in the following two subsections.

2.3.1. Hadamard Test

The Hadamard test is used to compute expectation values in the form of $\langle\psi|U|\psi\rangle$. The layout of the Hadamard test circuit is shown in fig. 2.2. In this figure it is assumed that $|\psi\rangle$ is already initialized by a unitary V, where $|\psi\rangle = V|0\rangle$. The quantum state after the second Hadamard-gate is $\frac{1}{2}\left(|0\rangle \otimes (I+U)|\psi\rangle + |1\rangle \otimes (I-U)|\psi\rangle\right)$. The probability to measure the zero and one state is shown below:

$$P(0) = \frac{1}{2}\left(1 + Re(\langle\psi|U|\psi\rangle)\right), \quad P(1) = \frac{1}{2}\left(1 - Re(\langle\psi|U|\psi\rangle)\right) \tag{2.7}$$

Subtracting these two values leads to $P(0) - P(1) = \langle\psi|U|\psi\rangle$ and yields the expectation value. On the right hand side in fig. 2.3, a more complex Hadamard test circuit is shown which is used in the VQLS algorithm, where the value resulting from $\langle 0|V^\dagger A^\dagger AV|0\rangle$ is computed [10]. Here the $S^\dagger$ gate as shown in the colored box is optional and depends on the amplitudes in the ansazt. When an ansatz only contains real valued amplitudes, the circuit without the S gate is used to evaluate the different terms. However, when complex amplitudes are present in the state prepared by the ansatz, more circuits have to be evaluated and the S-gate is required in order to evaluate the complex part contributing to the expectation value [10]. Since the Hadamard test is a probabilistic measurement, multiple samples are needed to obtain an accurate result. The accuracy scales with the number of measurements via $M_{tot} \propto 1/\epsilon^2$. Here M_{tot} is the total number of measurements or samples and ϵ the desired accuracy. This can be reduced to $M_{tot} \propto 1/\epsilon$, with a slight modification of the algorithm [5]. The Hadamard test works due to the principle of phase kickback, where the eigenvalues added by a controlled unitary are kicked back onto the control qubit.

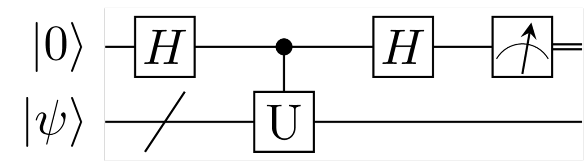


Figure 2.2: Hadamard test circuit used to compute $\langle\psi|U|\psi\rangle$. [85]

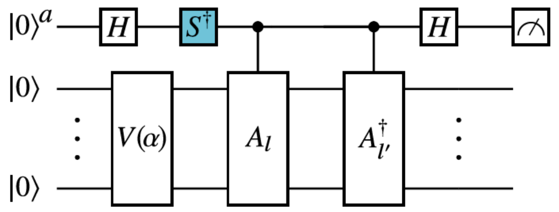


Figure 2.3: Hadamard test used to compute vector matrix products and inner products. [10]

2.3.2. Swap Test

The swap test was first shown in [13] and is a way to obtain the degree of overlap between two quantum states. This requires the input of two quantum states equal in size and a third register needed for the measurement of the output value. The circuit is shown in fig. 2.4. The probability of measuring 0 on the first qubit is:

$$P(0) = \frac{1}{2} + \frac{1}{2}|\langle\psi|\theta\rangle|^2 \tag{2.8}$$

[5]`https://quantumalgorithms.org/chapter-intro.html#modified-hadamard-test`

When the two quantum states are identical, the value for $|\langle\psi|\theta\rangle|^2$ is 1, and when they are far apart (orthogonal) it is zero. Computation of the square of the absolute value of the inner product between the states is the closest one can get to computing the actual inner product. It is impossible to compute the exact inner product itself on a quantum device since that would reveal the global phase of a quantum system. Since the measurement only yields a single value of either zero or one, multiple measurements have to be made to increase the accuracy of the swap test. The final value can then be computed with:

$$F = 1 - \frac{1}{M_{tot}} \sum_{i=1}^{P} M_i \tag{2.9}$$

Here F is the output value of the swap test, which is a measure of the fidelity, M_i is the number of measurements where 1 occurs and M_{tot} is the total number of measurements. The accuracy of this method scales inversely with the number of measurements $M_{tot} \propto 1/\epsilon^2$ [88].

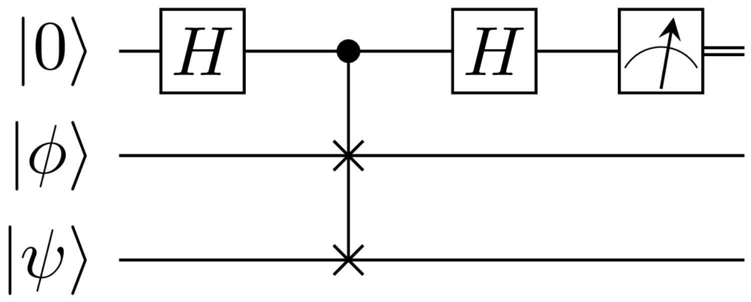


Figure 2.4: Swap test circuit used to compute $|\langle\phi|\psi\rangle|^2$. [86]

Even though the Hadamard test and the swap test look quite similar in what they compute and both rely on phase kickback, there are some differences. For the Hadamard test the unitaries creating the quantum state are assumed to be known, while for the swap test this is not the case. For the Hadamard test the implementation of the controlled-U gate can be non-trivial and can lead to complex circuits, which can be a downside of the implementation for the algorithm. For the swap test only a ladder of doubly controlled swap gates are needed for larger systems, where the control stays on the first qubit and the swapped qubits of both quantum states move to the next qubit for each gate. This means that n controlled swap gates are required, where n is the number of qubits of each quantum state.

2.4. HHL Algorithm

The HHL algorithm, named after its inventors Harrow, Hassidim and Lloyd [30], is an algorithm used to solve linear systems of equations. The method assumes that A is an Hermitian matrix, meaning that $A = A^\dagger = (A^*)^T$, where A^* is the complex conjugate of A. In case that A is not Hermitian, a simple trick can be used to make it solvable with the HHL algorithm. A new matrix $\tilde{A}$ can be defined as shown in eq. (2.10)

$$\tilde{A} = \begin{pmatrix} 0 & A \\ A^\dagger & 0 \end{pmatrix} \tag{2.10}$$

After which the system can be solved by using the three equations shown below.

$$\tilde{A} = \begin{pmatrix} 0 & A \\ A^\dagger & 0 \end{pmatrix} \quad (2.11) \qquad \tilde{A}\vec{y} = \begin{pmatrix} \vec{b} \\ 0 \end{pmatrix} \quad (2.12) \qquad y = \begin{pmatrix} 0 \\ \vec{x} \end{pmatrix} \quad (2.13)$$

Another important, and non-trivial, aspect of the HHL algorithm is the preparation of the state $|b\rangle$. In some cases, such as for an even distribution where only Hadamard gates are required, initialization of $|b\rangle$ can be very simple. However for other problems this is not the case, and in order to keep an advantage over classical methods the state must be efficiently prepared. Using quantum phase estimation, $|b\rangle$ is decomposed into the eigenvector basis and by using QFT and Hamiltonian simulation, the state $|x\rangle \approx A^{-1}|b\rangle$ is obtained. By using amplitude encoding, the solution vector $|x\rangle$ is encoded in the amplitudes in a normalized form. Using the full length of the quantum states allows for storing an exponentially increasing amount of data on a set of qubits. The HHL algorithm does have a non-zero probability of finding the ancilla qubit in the wrong state, in which case the algorithm has to be rerun.

For a sparse linear system and a low condition number κ, obtaining the the quantum state $|x\rangle$ (without measurements) scales with $\mathcal{O}\left(log(N)\kappa^2\right)$. κ is the condition number, which indicates how close a matrix is to being singular, and is defined as the ratio of the largest and smallest eigenvalues of matrix A. The fastest classical method for solving a sparse linear system is the conjugate gradient method has a run time of $\mathcal{O}(N\kappa)$ [6].

In [64] the complexity is stated as $\mathcal{O}\left(\kappa^2 s^2 \operatorname{poly}(\log(N), 1/\epsilon)\right)$, which is proportional to $\mathcal{O}\left(\kappa^2/\epsilon\right)$ when κ is the limiting factor. In recent years the complexity for solving linear systems has been improved and reduced to $\mathcal{O}(\kappa s \log(N) \log(1/\epsilon))$, however to achieve this complexity a completely different approach is used, which does not make use of phase estimation or amplitude amplification [43].

Even though it is a difficult task in itself to obtain a state proportional to $A^{-1}|b\rangle$, it does not solve the task of finding $|x\rangle$ efficient immediately. Since $|x\rangle$ is a quantum state it is not possible to directly extract all information out of it, and multiple measurements have to be made to obtain this. Since the quantum state of an n qubit problem, consists of $N = 2^n$ amplitudes, at least a number of quantum measurements proportional to N has to be made to reconstruct $|x\rangle$ [18]. This problem can be overcome, when not the entire reconstruction of $|x\rangle$ is required, and only a part of the vector is sufficient. For some problem classes, one is not interested in the exact vector x, but in an expectation value in the form $\langle x|M|x\rangle$, this way it is not needed to sample the solution state multiple times. Another solution would be to evaluate the inner product between $|x\rangle$ and a function r. A simple example is taking r to be uniform over a region, then $\langle r|x\rangle$ gives the average of $|x\rangle$ over that region [53]. This inner product can be found by performing a Hadamard test [2].

With the problem of sampling the solution, one might question whether the HHL algorithm has any use at all compared to classical methods, but as A. Childs states it [18]: "In particular, they show that any quantum computation can be encoded into an instance of solving linear equations, even with the restrictions required for their quantum solver to be efficient.". From this statement, the conclusion is made that the HHL algorithm is not efficiently simulated on classical computers. Since if it would be possible to efficiently simulate HHL classically, it means that all quantum algorithms could be efficiently simulated classically. And thus the algorithm can be used to solve non-trivial problems.

Solving the Poisson equation by means of HHL has been performed by Morrell et al. [54]. In this work, attempts at finding a solution for a 2-qubit problem on an IBM quantum computer were made. However, due to the low gate fidelity of the hardware, obtaining accurate results is still difficult. Furthermore, this study also states that fidelity is not a good metric to indicate accuracy of the solution. Due to the large circuit depth of the HHL algorithm and low gate fidelity in NISQ devices, different methods have to be investigate for near term successes.

2.5. VQLS Algorithm Overview

The variational counterpart of the HHL algorithm, is the variational quantum linear solver (VQLS). It was devised in 2019 by Bravo-Prieto et al. [10], and is a hybrid quantum-classical algorithm. It is constructed with the aim to solve linear systems of equations on NISQ devices. As the name implies it is a linear solver and variationally finds the state $|x\rangle$ such that $A|x\rangle \propto |b\rangle$, where $|x\rangle = x/||x||$ is the normalized vector x. In this section the general working and overview of VQLS will be given.

The basic layout of the algorithm is similar to that of any other variational algorithm. The cost function is evaluated on a quantum computer, while the optimization for the parameters happens on a classical computer. Like other VQAs, it also starts with a parameterized ansatz $V(\alpha)|0\rangle = |\psi\rangle$, which often uses randomized input parameters. When the input parameters are optimized the output of the quantum

[6]This can depend on the details, such as sparseness of the matrix that is being solved.

state is $V(\alpha_{opt})|0\rangle = |x\rangle$, where V is the ansatz operation and the $|x\rangle$ the problem solution. The layout of the algorithm is shown in fig. 2.5.

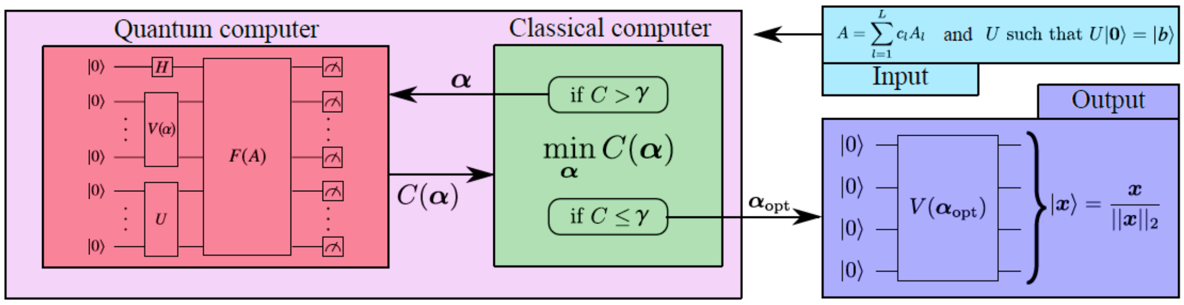


Figure 2.5: Layout of the VQLS algorithm. [10]

From the figure, it becomes clear that the algorithm consists of three main components: the input, the output and the evaluation of the cost function. The input consists of the matrix A and the unitary U, which forms $U|0\rangle = |b\rangle$. For simple cases of $|b\rangle$, this can be a simple circuit, but for more complex cases another parameterized ansatz is required. Since GBQC requires unitary operations to act on the qubits, it is not possible to just feed any arbitrary matrix A to the algorithm. It first has to be decomposed into a linear combinations of unitaries (LCU), such that $A = \sum_{l=1}^{L} c_l A_l$, where c_l are complex coefficients and A_l are the unitary operators. Often times the unitaries are decomposed into the Pauli matrices $\sigma_{x,y,z}$. The block $F(A)$, is a set of quantum operations used to evaluate the different terms in the decomposition of A. For each different term in the decomposition, $F(A)$ can contain different gates, so different circuits have to be executed. Finally, the output is obtained in the form of a quantum state, generated by the optimized ansatz such that $V(\alpha_{opt})|0\rangle = |x\rangle$. The authors of the original VQLS paper assume that an efficient decomposition of A exists, that the condition number of A is finite and that $||A|| \leq 1$ also must hold[7]. For sparse matrices, efficient decomposition into the Pauli matrices is guaranteed, but for non-sparse matrices this is more complicated.

The next part of the algorithm is the evaluation of the cost function $C(\theta)$. VQLS uses an ansatz that prepares a quantum state $U(\theta)|0\rangle = |x\rangle$, which is initialized by the parameter(s) θ. The cost function should be formulated such that the optimal angles that obtain a minimal cost, result in $U(\theta_{opt})|0\rangle = |x\rangle \propto x$. A simple example of a cost function used in the VQLS, is the global effective Hamiltonian cost function. By defining the Hamiltonian H as eq. (2.14), the cost function is exactly zero when $|\psi\rangle \propto A^{-1}|b\rangle$. With this Hamiltonian the cost can be defined as in eq. (2.15).

$$H = A^{\dagger}\left(I - |b\rangle\langle b|\right)A = A^{\dagger}A - A^{\dagger}(|b\rangle\langle b|)A \tag{2.14}$$

$$C = \langle\psi|H|\psi\rangle = \langle 0|U(\theta)^{\dagger}|H|U(\theta)|0\rangle \tag{2.15}$$

This is a non-normalized cost function which can have large values far away from the solution and is exactly zero at the solution. The cost function $C(\theta)$ is evaluated by measuring the expectation value of the decomposed matrix A and the prepared quantum state. The different types of cost functions and how one can evaluate them is further discusses in section 2.7.

Once a cost function has been selected, the only remaining part is the classical optimizer. This optimizer is used to vary the input parameters θ such that $C(\theta)$ is minimized. The presence of an efficient decomposition is a very important aspect of the VQLS algorithm, as during each optimization iteration all the terms in the decomposition of the matrices need to be evaluated. The accuracy of the evaluation of the expectation value scales with the number of times a circuit is run, which is also called the number of shots. Based on the required accuracy of the evaluation of the expectation value and the total number of terms this can quickly lead to an inefficient algorithm.

By varying and optimizing the input parameters θ by means of a classical optimizer, the quantum state that is proportional to the solution $|x\rangle$ is obtained. The type of cost function can greatly affect

[7]The paper does not specify which norm is referred to as $||A||$.

the success of the algorithm and is covered in more detail in section 2.7. More about the classical optimizers is elaborated in section 2.10.

2.6. Ansatze

The first aspect of the VQLS algorithm that will be discussed in more detail is the ansatz. An ansatz corresponds to an educated guess or assumption. In variational algorithms the ansatz is used to prepare a quantum state with which the expectation value of a certain operator is computed, and which finally should correspond to the correct solution. Two important aspects of an ansatz are the expressive power and the level of entanglement. The expressive power refers to the amount of the Hilbert space that an ansatz can span. A short and simple ansatz with few parameters might be more feasible to realize on NISQ devices, but if the ansatz does not span the right areas of the Hilbert space, it might not contain the solution one is looking for. When increasing the complexity of the ansatz and increasing the number of parameters to optimize, the full Hilbert space can become available, but in the meantime optimization can also become increasingly difficult to a point where it is not possible to get close to a solution in sub-exponential time.

Ansatze are comparable to classical neural networks in the sense that parameters are optimized in order to solve problems. Du et al. [21] state that even a simple structured parameterized quantum circuit can outperform any classical neural network for generative tasks, indicating that variational algorithms might be a good direction for quantum supremacy on NISQ devices.

The construction of a good ansatz is not a straightforward process. Some ansatze might have relative poor expressibility for a single layer, but might improve considerably when more layers are added. This way a very expressible circuit will not necessarily be better than a shallow circuit, as the performance with multiple layers also has to be considered. In the study conducted by Sim et al. [77], the expressibility of 19 ansatz circuits has been tested. Here a maximum of 5 layers is considered per circuit. They showed that the expressibility of different ansatze, saturate at different levels. This is to be expected since shallow ansatze are not expected to explore the full Hilbert space. In addition, the number of layers required to reach saturation is also different depending on the ansatz. The rate at which saturation changes as more layers are added is also affected by the ansatz structure. One notable thing from this study is that not all circuits have increased expressibility for increasing number of layers. As a measure of expressibility this study uses the Kullback-Leibler (KL) divergence. Which is a measure of how two probability distributions differ from each other. It is often used in machine learning between the estimated fidelity distribution and that of the Haar distributed ensembles [77]

In the subsections below, the two most used ansatze are explained. These are the Hardware Efficient Ansatz and the Quantum Alternating Operator Ansatz. Other well known and promising ansatze are the dynamic ansatz [60], the ansatz tree approach [33], the alternating layered and tensor product ansatz [55], the unitary coupled cluster ansatz [5, 66, 74] and the Adaptive Derivative-Assembled Pseudo-Trotter ansatz Variational Quantum Eigensolver (ADAPT-VQE) [27]. However, since these last ansatze play no role in the current work, they are left out of the literature review.

2.6.1. Hardware Efficient Ansatz

A popular ansatz on NISQ devices is the hardware efficient ansatz (HEA). It is a basic problem agnostic ansatz that makes use of the available gates that are native to a certain quantum machine. The HEA consists of alternating blocks of single qubit rotation gates and two-qubit entangling gates. By using more layers the expressibility of the ansatz can be increased. The term Hardware Efficient Ansatz was first introduced by Kandala et al. [37] for a variational quantum eigensolver and they used eq. (2.16) for their mathematical definition of HEA.

$$|\Phi(\vec{\theta})\rangle = \prod_{q=1}^{N}\left[U^{q,d}(\vec{\theta})\right] \times U_{\mathrm{ENT}} \times \prod_{q=1}^{N}\left[U^{q,d-1}(\vec{\theta})\right] \cdots \times U_{\mathrm{ENT}} \times \prod_{q=1}^{N}\left[U^{q,0}(\vec{\theta})\right] |00 \ldots 0\rangle \qquad (2.16)$$

In this equation $U^{q,d}$ specifies one of the Pauli gates with N the total number of qubits, q the qubit position and d the respective layer, U_{ent} is a general term for the entangling block. An example of what the HEA looks like is shown in fig. 2.6.

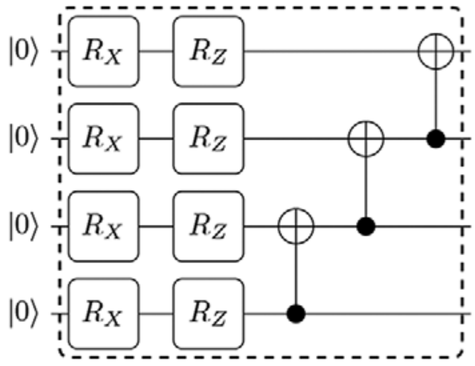


Figure 2.6: Simple example of a hardware efficient ansatz. [77]

Here the ansatz is shown for a single layer consisting of R_X and R_Z-gates and an entangling layer consisting of CNOT gates. This is a very simple case of the HEA, and by using more complex structures and rotation gates the expressibility of the ansatz can be greatly increased. Another point here is that the number of parameters for one ansatz structure can also differ greatly. For example, in fig. 2.6 the rotation gates can be initialized with 2 parameters in total, one for the R_X gates and one for the R_Z gates. But it is also possible to use 8 parameters in total such that each gate is controlled individually. The latter will have greater expressibility, but also will be more difficult to optimize, since the number of parameters is 4 times as high.

Another method to construct an HEA is by using alternating blocks as shown in fig. 2.7

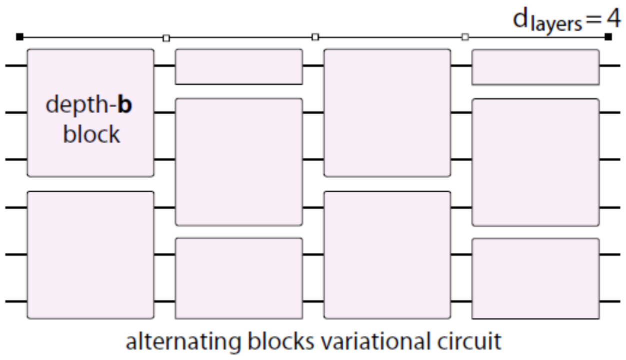


Figure 2.7: Alternating block structure for a 6-qubit circuit with 4 layers. [40]

By using alternating blocks, the qubits within each block are entangled locally. As will be explained in section 2.7, a local cost function has beneficial properties regarding the performance of the optimizer [16, 17]. By using smaller blocks the trainability of the ansatz is increased, while the expressibility stays unchanged [40, 55].

In short, the general idea behind HEA and its alternating form is to keep the circuit shallow, as on NISQ devices gate fidelity is relatively low, while still having a sufficient level of expressibility to obtain the correct results. Again it has to be emphasized that HEA does not have one specific structure and can range from very simple structures ready to implement on NISQ devices, to very large and deep circuits.

2.6.2. Quantum Alternating Operator Ansatz

The Quantum Alternating Operator Ansatz, also known as QAOA, is a more complex form of ansatz. This ansatz is based on the Quantum Approximate Optimization Algorithm (which is also abbreviated as QAOA) as introduced by Farhi and Goldstone in 2014 [23]. As opposed to the problem agnostic HEA, this ansatz takes into account characteristics of the problem it is solving. The ansatz consists of alternating layers between a driver Hamiltonian H_D and a mixer Hamiltonian H_M and by extending the number of layers consisting of the driver and mixer unitaries the approximation of the solution improves. QAOA is universal as the number of circuit layers tends to infinity and stands out as an efficient ansatz, since it reduces the size of the solution space that is available to the qubits [23]. QAOA can mathematically be described by:

$$U(\theta) = U_M\left(\theta_l^L\right) U_D\left(\theta_{l-1}^L\right) \cdots U_M\left(\theta_2^1\right) U_D\left(\theta_1^1\right) \tag{2.17}$$

Where the mixer unitary is represented by $U_M(\theta_p^q) = e^{-i\theta_p^q H_M}$, and the driver unitary $U_D(\theta_p^q) = e^{-i\theta_p^q H_D}$. The driver and mixer Hamiltonians depend on the problem. Often times the mixer Hamiltonian is a simple form such as the Pauli-X gate or another form of rotation gates applied to all qubits. The driver Hamiltonian is more complex and takes into account details of the objective function that is being solved. Initially QAOA is brought into superposition by means of the Hadamard gate, by applying it to all qubits. Once the whole system is in a superposition, the layers of driver and mixer Hamiltonians are applied.

QAOA originally started as a method for combinatorial optimization problems, but can also be applied as an ansatz to VQAs. Hadfield et al. [28] states that the potential of QAOA should be investigated and that its effectiveness as an heuristical method to obtain solutions must be further explored. QAOA has been used to solve MaxCut problems [83] and for solving the Poisson equation by means of VQLS [44]. The main difficulty of using QAOA is forming the driver Hamiltonian. In [83] the Jordan-Wigner transformation is used as the driver, while in [44] no motivation for the specific driver is given.

In the paper by Liu et al. [44], which plays an important role in this thesis, the QAOA ansatz is used. Here they implemented eq. (2.18) for the mixer unitary, where X_j is the Pauli X-operator on the j-th qubit. m refers to the total number of qubits in the system. For the driver unitary they used eq. (2.19), here Y and Z are the Pauli Y and Z operators. In order to implement this a combination R_z, R_x and CNOT-gates are needed. The ansatz depth scales with $1 + (3n + 6)L$, where n is the number of qubits and L the number of layers.

$$U_M = \sum_{j=0}^{m-1} X_j \tag{2.18}$$

$$U_D = \sum_{j=0}^{m-1} Z_j Z_{j+1} + Z_{m-1} Z_0 + Y_0 Y_1 \tag{2.19}$$

2.7. Cost Function

Cost functions are often used in optimization problems to evaluate the cost of a certain event. In terms of variational quantum algorithms, cost functions are used to minimize the expectation value of a Hamiltonian in order to obtain the solution to a problem. An example of this in quantum chemistry is the lowest energy state of a system corresponding to the lowest eigenstate of a Hamiltonian. The minimization of a cost function C initialized by a parameter or vector θ can be described as:

$$\theta_{opt} = \arg\min_{\theta} C(\theta) \tag{2.20}$$

Here the optimizer iterates over a scalar or vector θ such that the cost function is minimized. It should be noted that a cost function can also be configured such that the optimal solution is obtained at a maximum, however in almost all quantum application a minimization procedure is used. For optimization of QUBO or Ising models a cost function with a similar form is used. For these problems also the expectation value of a Hamiltonian is minimized.

Using a cost function is not just limited to problems in the shape of a QUBO or Ising model, or solving linear systems, where a Hamiltonian is minimized. Different problems can be solved with different forms of cost functions. For example, different cost functions, such as one shown in eq. (2.21), are used in variational quantum circuits to perform singular value decomposition, which is a method in linear algebra for the factorization of a matrix [11, 82]. In this equation d_H is the Hamming distance between two measurement sets $M_j^{A,B}$ and can also be expressed in terms of two-local Pauli σ_z operators. The

Hamming distance is a measure for how close two bit strings of data are. $\sigma_z^{q,A}$ and $\sigma_z^{q,B}$ are the Pauli Z operator on qubit q for set A and B respectively.

$$C \equiv \sum_j d_H\left(M_j^A, M_j^B\right) \equiv \sum_q \frac{1 - \left\langle \sigma_z^{q,A} \sigma_z^{q,B} \right\rangle}{2} \tag{2.21}$$

2.7.1. Types of Cost Functions

Various different approaches can be used for the cost function when solving a linear system. One of the most simple types of cost functions is obtained by defining a Hamiltonian, where the ground state corresponds to the solution of the problem at hand. The cost function can then be expressed as shown in eq. (2.22). Where $\hat{C}$ is a non-normalized cost function, H is the problem Hamiltonian and $|\psi\rangle$ is an arbitrary quantum state, often generated with a parameterized ansatz such as $U(\theta)|0\rangle = |\psi\rangle$.

$$\hat{C} = \langle\psi|H|\psi\rangle \tag{2.22}$$

A variety of problems can be solved with this equation, such as finding the ground state energy of a molecule or solving QUBO problems. For VQLS the effective Hamiltonian is a simple way to obtain the solution by means of optimizing for the expectation value. To do this the Hamiltonian H in eq. (2.22) must be in the form of:

$$H = A^\dagger\left(I - |b\rangle\langle b|\right)A = A^\dagger A - A^\dagger(|b\rangle\langle b|)A \tag{2.23}$$

It can be shown that H is a semi-definite matrix, which has a minimum eigenvalue of zero, and the corresponding eigenstate $|x\rangle$ is the solution vector. Since A is invertible this also means that $|x\rangle$ is the unique eigenstate. Thus finding the ground state of this Hamiltonian corresponds to finding the only solution of a linear system $Ax = b$ [44]. Combining equation eq. (2.22) and eq. (2.23) makes it more clear which terms have to be evaluated and results in:

$$\hat{C} = \langle\psi|A^\dagger A|\psi\rangle - \langle\psi|A^\dagger(|b\rangle\langle b|)A|\psi\rangle = \langle\psi|A^2|\psi\rangle - |\langle b|A|\psi\rangle|^2 \tag{2.24}$$

In this equation the right hand side is only valid if A is a real matrix, such that $A^\dagger A = A^2$, which is the case for the discretized Poisson equation. Here $|\psi\rangle \propto A|x\rangle$ holds [8].

The downside of a cost function as formulated above, is that it can contain local minima. The resulting value will not only be minimal when $|\psi\rangle$ corresponds to the eigenvector of the lowest eigenvalue, but also when the norm of $A|\psi\rangle$ is small , meaning that local minima can occur which makes optimization of the cost function more difficult. One method to prevent the above phenomenon is by using a normalized cost function as shown below:

$$C = \frac{\langle\psi|H|\psi\rangle}{\langle\psi|A^2\psi\rangle} = 1 - \frac{|\langle b|A|\psi\rangle|^2}{\langle\psi|A^2|\psi\rangle} \tag{2.25}$$

This way the cost function will only approach zero when the prepared state is actually close to the solution vector $|b\rangle$. However, this cost function comes with other problems. It is a so called global cost function and when combined with random initialization of the ansatz parameters, it might have problematic convergence during optimization and lead to a phenomenon named barren plateau. When this happens, the variance of the cost function gradient vanishes exponentially in the number of qubits. To circumvent this Bravo-Prieto et al. [10] proposed the use of local cost functions. Local cost functions make use of the local properties of the qubits. The local cost functions they propose are shown in eq. (2.26).

$$\hat{C}_L = \langle\psi|H_L|\psi\rangle \tag{2.26}$$

Where H_L is the effective local Hamiltonian:

[8] It should be noted that for a decomposed non-unitary matrix, $|\psi\rangle = A|x\rangle$ is not valid. Only for a unitary A this will hold. Otherwise it will be $|\psi\rangle = cA|x\rangle$, where c is a constant.

$$H_L = A^\dagger U \left(\mathbb{1} - \frac{1}{n} \sum_{j=1}^{n} |0_j\rangle\langle 0_j| \otimes \mathbb{1}_{\bar{j}} \right) U^\dagger A \tag{2.27}$$

In this equation $|0_j\rangle\langle 0_j|$ is the zero state on qubit j and $\mathbb{1}_{\bar{j}}$ is the identity on all qubits except j. In the paper by Bravo-Prieto it is proven that the local cost function and global cost function both converge to zero for the same ansatz parameters. So if a certain vector θ results in a cost value close to zero for the global cost function, it will also do so for the local one and vice versa. Again it is also possible to normalize the local cost function with $C_L = \hat{C}_L / \langle\psi|A^2|\psi\rangle$, in order to improve the cost function landscape. The local cost function has been used to optimize systems with up to 50 qubits, where global cost functions could become difficult to optimize due to the vanishing gradients (barren plateaus). [10]

2.7.2. Evaluation of Cost Functions

The evaluation of a cost function is an important part in most VQAs, since this is the part where speedup over classical methods takes place. The optimizer makes use of a classical algorithm and post-processing of data also happens on a classical device, so no speedup takes place in these parts of the algorithms. This means that evaluating the cost function must be done efficiently. Evaluating the expectation value of an operator can be done in multiple ways. For some problems it can be as simple as evaluating one expectation value between a unitary matrix and a single quantum state. For more difficult problems, such as what is found with the Poisson equation, a linear decomposition of unitaries might be required. In order to construct the problem Hamiltonian, such that an arbitrary matrix A can be decomposed as $A = \sum_{l=1}^{L} c_l A_l$, where c_l are complex coefficients and A_l are unitary operators consisting of a tensor product of quantum operators. Often the Pauli matrices $\sigma_{x,y,z}$ are used for such a decomposition. The expectation value for each individual unitary operator A_l can then be measured, multiplied by their respective coefficient and summed to obtain the expectation value for the complete matrix A.

The expectation value is the expected value resulting from an operator acting on a quantum state, and can be obtained in multiple ways. An example of an expectation value in a cost function is $\langle b|A|\psi\rangle$ as shown in eq. (2.25) and its computation is done as shown in the equation below [41].

$$\langle b|A|\psi\rangle = \langle 0|B^\dagger A U(\theta)|0\rangle = \sum_{l=1}^{L} c_l \langle 0|B^\dagger A_l U(\theta)|0\rangle \tag{2.28}$$

In the above equation B is the operator to generate the quantum state proportional to b, A and A_l are the problem matrix and its decomposed form respectively, and finally $U(\theta)$ is the ansatz operator. By minimizing the cost, the quantum state that is proportional to the solution can be found: $|\psi\rangle = U(\theta)|0\rangle \propto x/||x||$. The expectation value can be computed via the Hadamard test as explained in section 2.3. Due to the complexity of implementing a controlled-unitary for larger systems, the Hadamard-test can be difficult to implement on NISQ devices. Another method, that omits the use of the Hadamard test, is the effective Hamiltonian approach [41, 44]. This method first defines an effective Hamiltonian as shown in eq. (2.23), of which the ground state corresponds to the normalized solution of $x = A^{-1}b$.

It can be shown that H is a semi-definite matrix, which has a minimum eigenvalue of zero, and the corresponding eigenstate is $|x\rangle$, the solution vector. Since A is invertible this also means that $|x\rangle$ is the unique eigenstate [44]. As mentioned above, this method does not require the Hadamard test, but needs to decompose the right hand side of eq. (2.23). This can be done by using eq. (2.29) and eq. (2.30). By using a Pauli decomposition and measuring the qubits in the basis of each respective Pauli term the expectation value of $\langle\psi|A_l|\psi\rangle$ can be computed without using the Hadamard test. Both equations require $O(L^2)$ terms to be evaluated, meaning that again an efficient decomposition of the unitaries is important for the speedup over classical algorithms.

$$A^\dagger A = \sum_{l=1}^{L} |c_l|^2 I + 2 \sum_{k=1}^{L} \sum_{1=l<k}^{L-1} \mathrm{Re}\left[c_l^* c_k\right] A_l A_k \tag{2.29}$$

$$A^\dagger(B|0\rangle\langle 0|B^\dagger)A = \sum_{k=1}^{L}\sum_{l=1}^{L} c_k^* c_l A_k (B|0\rangle\langle 0|B^\dagger) A_l \tag{2.30}$$

Downsides of this approach are that it can lead to many terms in the decomposition, but this can also partly be circumvented by using simultaneous measurement of observables during computation of the expectation value.

On a final note, the cost evaluation of a quantum circuit in terms of two-qubit gates is not as simple as it looks. On different quantum machines different native gates are used, and transpilation of the two-qubit gates into native gates can result in different results depending on the machine. Solving this problem is not the goal of the thesis, but is something to keep in mind when considering the overall efficiency of VQLS.

2.8. Matrix Decomposition

The previous section on the evaluation of the cost function showed that computing the expectation value of A and the quantum states $|b\rangle$ and $|x\rangle$ is required. In the case where A is unitary, this is a relatively simple process, since these matrices can be directly measured on a quantum computer. However, in the case where A is not-unitary, such as is the case for the discretized Poisson equation, a different approach is needed. By decomposing A into a linear combinations of unitaries (LCU) it is possible to measure all the unitary gates individually and then multiply them by their respective coefficient to obtain the final value. This decomposition can be written as $A = \sum_{l=1}^{L} c_l A_l$. Here L indicates the number of terms in the decomposition, c_l is the (complex) coefficient of the l-th term and A_l is a unitary matrix of the l-th term. The number of terms L that form the matrix should scale logarithmically in problem size and A_l should be constructed from operators that are simple to measure on a quantum device.

With the normalized cost function as in eq. (2.25), the term $\langle\psi|A^2|\psi\rangle$ has to be evaluated by eq. (2.31), which consist of $L(L-1)/2 = \mathcal{O}(L^2)$ different terms [10]. This means that if the number of terms L in the decomposition of A scales quadratically, the total number of terms scales exponentially and all hope for speedup over classical methods is lost. So a decomposition that scales sub-quadratically in the number of qubits is required.

$$\langle\psi|A^2|\psi\rangle = \sum_{ll'} c_l c_{l'} \langle 0|V^\dagger A_{l'}^\dagger A_l V|0\rangle \tag{2.31}$$

As mentioned in chapter 1 the Poisson equation will be solved with Dirichlet boundary conditions. By using second-order central difference discretization, the matrix A results in eq. (2.32). So a suitable decomposition should be found, which can exactly reconstruct this matrix.

$$A = \begin{bmatrix} 2 & -1 & 0 & \dots & & \\ -1 & 2 & -1 & 0 & \dots & \\ 0 & -1 & 2 & -1 & & \\ \vdots & & \ddots & \ddots & \ddots & \\ & & & -1 & 2 & -1 \\ 0 & 0 & 0 & \dots & -1 & 2 \end{bmatrix} \tag{2.32}$$

2.8.1. Pauli Decomposition

A suitable set of unitaries to first attempt to decompose the Poisson equation would be the Pauli matrices. By using a linear combination of operator tensor product in the set $\{I, X, Y, Z\}$, it is possible to reconstruct the A-matrix. For the 2-qubit case ($N = 4$), the A matrix is formed by $A = 2II - 1IX - 0.5XX - 0.5YY$, resulting in 4 different terms. In this notation IX is short for the tensor product $I \otimes X$, where I is the identity operator on the first qubit and X the Pauli-X gate on the second qubit. As the problem size increases, more and more terms are required and it is possible to express this decomposition in a recursive equation. However, the scaling of the number of terms is exponential in the number of qubits, meaning that a cost function evaluation is quadratic in the number of unitaries decomposing A. Neglecting the small terms with this decomposition is not a suitable method to lower the number of terms, since these terms are important for the coupling of the whole linear system. This means that using a Pauli-decomposition will not be a suitable method for solving the Poisson equation by means of VQLS with the aim of achieving speedup over classical methods. [14]

2.8.2. Raising and Lowering Operator Decomposition

A different method of decomposition is introduced by Liu et al. [44], in this work the Poisson equation with Dirichlet boundary conditions is decomposed into raising and lowering operators. They make use of the effective Hamiltonian approach as eq. (2.24) shows. This way they require the decomposition of A and A^2, where A is equal to the discretized form of the Poisson equation shown above in eq. (2.32). Instead of using the unitary Pauli matrices for the decomposition, they use the raising and lower operators $\sigma_+ = |0\rangle\langle 1|$ and $\sigma_- = |1\rangle\langle 0|$. With these operators matrix A is expressed for the single qubit case as $A = 2I - \sigma_+ - \sigma_-$. For the two qubit case, the matrix can be expressed as $A = IA_1 - \sigma_-\sigma_+ - \sigma_+\sigma_-$. This can be put into a recursive equation as shown in eq. (2.33). This results in $2m + 1$ terms to describe a matrix of size 2^m (m-qubit problem).

$$A_m = I \otimes A_{m-1} - \sigma_- \otimes \underbrace{\sigma_+ \otimes \cdots \otimes \sigma_+}_{m-1} - \sigma_+ \otimes \underbrace{\sigma_- \otimes \cdots \otimes \sigma_-}_{m-1} \tag{2.33}$$

For the A^2 matrix the same approach as described above can directly be used, but results in $(2m+1)^2$ terms, which is inefficient compared to classical methods. By using $A^2 = B - C$, an efficient decomposition for A^2 is obtained. Here C is zero apart from its first and last entry, where it has a value of one. B is the remaining part, such that the matrix A^2 is formed when C is subtracted. With this approach the number of terms is reduced to $4m - 1$ for B and 2 for C, resulting in $4m + 1$ terms for A^2. The total decomposition of both A and A^2 results in $6m + 2$ terms, which is sub-exponential. The decomposition also works for higher dimensions of the Poisson equation and can be extended to Neumann and Robin boundary conditions for the 2D case.

By using the raising and lower operators to express matrix A another problem is introduced. The Pauli matrices are unitary and can easily be measured in the correct basis to obtain the expectation value of each tensor product of unitaries. The raising and lowering operators are not unitary, and thus cannot be measured this way. To still be able to measure them, special observables are used to compute each tensor product in the decomposition. For this, the Bell states are used to construct quantum states in which measurements can directly be performed to obtain the expectation value $\langle\psi|H|\psi\rangle$. Rotating the qubits into the correct basis requires at most $m - 1$ CNOT-gates for an m-qubit sized problem. Which means, that the depth of the circuits is hardly affected by this approach and makes it suitable for implementation on a NISQ device. These measurements can be performed with the Hadamard test and are explained in more detail in section 3.5. [44]

2.8.3. Shift Operator Decomposition

The decomposition into raising and lowering operators is already much more efficient than the Pauli decomposition for the Poisson equation. However, the study by Sato et al. [69] introduced an even more efficient decomposition. By introducing the shift operator they are able to decompose the Poisson equation in $\mathcal{O}(1)$ terms. Meaning that when the problem size is increased the number of terms in the decomposition remains constant. With this operator, the Poisson equation can be decomposed with Dirichlet, Neumann and periodic boundary conditions. Furthermore, it can also be extended to higher dimensional problems. The shift operator P, entangles all qubits by means of multi-controlled gates as is shown in fig. 2.8.

Mathematically this operator is defined as:

$$P = \sum_{i \in [0, 2^n - 1]} |(i+1)\, mod 2^n\rangle\langle i| \tag{2.34}$$

The A-matrix with Dirichlet boundary conditions can then be formed by eq. (2.35), where $A_{periodic}$ is defined as eq. (2.36).

$$A_{Dirichlet} = A_{periodic} + P^{-1}\left(I^{\otimes n-1} \otimes X\right)P \tag{2.35}$$

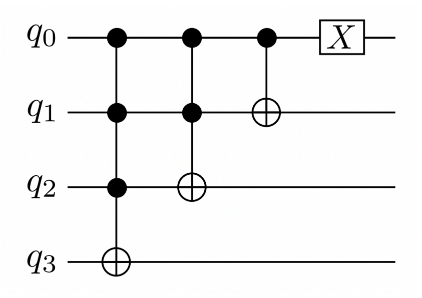


Figure 2.8: Quantum circuit for the implementation of the shift operator on 4 qubits. [69]

$$A_{periodic} = I^{\otimes n-1} \otimes (I - X) + P^{-1}\left(I^{\otimes n-1} \otimes (I - X)\right)P \tag{2.36}$$

Since in these equations the problem size n is only present inside P and in the tensor products of the identity operator, it becomes clear that the decomposition is indeed independent of the problem size. However, this does not mean that the shift operator is the perfect solution. When looking at fig. 2.8, to see what the operator P actually represents in a circuit, it becomes clear that there are a lot of controlled gates. The first one of these even entangles all qubits in the system. This immediately results in two problems and forms a major bottleneck for the implementation on a NISQ device. The first problem is that the implementation of multi-controlled gates is still a very difficult task. These gates can be seen as k-controlled Toffoli gates, which can be decomposed into easier implementable relative-phase Toffoli gates. However, doing so still leads to a circuit depth of $\mathcal{O}(n^2)$ and $2n - 3$ qubits. For NISQ devices this quickly becomes a problematic depth. [69]

The second problem is that the shift operator is a global operator as it entangles all qubits individually. As mentioned in section 2.7 global operators suffer from barren plateaus, which will lead to a difficult time during optimization of the ansatz parameters. Because of these reasons this implementation is not considered during this thesis work.

2.9. Barren Plateaus

One of the challenges regarding parameterized quantum circuits and variational quantum algorithms, is the phenomenon of barren plateaus. Barren plateaus were first noticed by McClean et al. [50], and they describe barren plateaus as an exponentially small probability that the gradient along any reasonable direction is non-zero as a function of the number of qubits. As a result, optimization of the ansatz parameters becomes increasingly difficult. Most optimizers are iterative solvers that are gradient based, meaning that the parameters for the next iteration are based on the gradient obtained from previous iterations. The first idea to resolve this issue would be to resort to gradient free optimizers, but even these optimizers do not rule out the occurrence of barren plateaus [6]. Barren plateaus are a phenomenon in all VQAs, but also in other applications such as quantum neural networks (QNN) [50].

Barren plateaus can also be described as a vanishing variance of the gradient such as shown in eq. (2.37) [70]. Here $\partial_\theta C$ refers to the gradient of the cost function C with respect to a parameter θ and $\langle\rangle$ takes the mean values of these terms. An example of a parameter θ is the rotation angle for a gate in the ansatz.

$$\mathrm{Var}\left[\partial_\theta C\right] = \left\langle (\partial_\theta C)^2 \right\rangle - \langle \partial_\theta C \rangle^2 \tag{2.37}$$

For quantum neural networks, which have great similarities with the VQLS algorithm, it has been shown that higher-order derivatives also vanish exponentially in the number of qubits [15]. A vanishing variance of the gradient does not necessarily mean that optimization becomes impossible, but often it does results in the fact that it becomes infeasible. With the vanishing gradient, optimization becomes more expensive and the accuracy of the measurements more important, and this is where another problem comes in. With the devices available in the NISQ era, quantum circuits and their measurements are not yet perfect, and with the number of measurements required for a certain accuracy ϵ scaling with $\mathcal{O}(1/\epsilon^2)$, this means that the number of required measurements becomes increasingly large. [70]

2.9.1. Haar Measure & Unitary t-Designs

An important aspect in the occurrence of barren plateaus are unitary-t designs. Unitary t-designs describe the distribution of a quantum state over the Haar measure. Before this can properly be understood, first the Haar measure itself must be explained. As the name implies, Haar measure is part of measure theory, which looks at how things are distributed. An example of using measure theory is obtaining the volume of a sphere by integration. Points on a sphere can be described by using 3 parameters: the radial distance ρ, the polar angle θ and the azimuthal angle ϕ. However, when one tries to simply integrate over the range of these parameters to obtain the total volume, the resulting value is incorrect. By adding the ”measure” to the integral, as is shown in eq. (2.38), the correct value for the volume is obtained.

$$V = \int_0^r \int_0^{2\pi} \int_0^{\pi} d\rho d\phi d\theta = 2\pi^2 r \longrightarrow V = \int_0^r \int_0^{2\pi} \int_0^{\pi} \rho^2 \sin\theta d\rho d\phi d\theta = \frac{4}{3}\pi r^3 \tag{2.38}$$

This step of adding the ”measure” is an important part when sampling a space uniformly at random. In the case of the volume integral, too much weight is put around the equator without including the measure. When sampling a quantum state, a similar principle can be applied as single-qubit states are nicely represented on the Bloch sphere. Again, by using this method the quantum space can be sampled truly uniformly at random.

For an $N = 2^n$ dimensional system, where n is the number of qubits, the unitary matrix acting on it forms the unitary group $U(N)$. Operations such as integration, applying a function or sampling can be performed over them. Similarly to the example of obtaining the volume of a sphere by adding the measure to an integral, to uniformly sample a quantum state the same principle can be used. This is where the Haar measure comes in and is used when working with unitary groups. The Haar measure for an integral can be expressed in the form of eq. (2.39), where $d\mu_N(V)$ can be decomposed into multiple components that contribute to the measure. Here $f(V)$ is a function that acts on $U(N)$ and $\mu_N(V)$ is the Haar measure. [62]

$$\int_{V \in U(N)} f(V) d\mu_N(V) \tag{2.39}$$

In 2 dimensions (single qubit case) it can be shown that in order to truly randomly sample the Bloch sphere, one needs to add the Haar measure. If this term is not taken into account, the sampling will be non-uniformly and weighted more around the poles of the sphere. As the number of qubits in a problem increase, the number of dimensions of the unitary also increase and thus the degree of concentration of the measure will increase. When randomly sampling in higher dimensions, this has the effect of making different samples in a randomly sampled state look very similar. Levy’s lemma shows that for a function $f(x)$, the probability that a sample deviates from the mean $E[f]$ is given by:

$$\Pr(|f(x) - E[f]| \geq \epsilon) \leq 2\exp\left[-\frac{N\epsilon^2}{9\pi^3\eta^2}\right] \tag{2.40}$$

Here f is again some function, ϵ is an arbitrary margin, N is the dimensionality of the problem and η is the Lipschitz constant. This shows that the probability of finding a sample that deviates significantly from the mean of a function reduces exponentially [62]. This is where barren plateaus enter the problem. So the closer one is to having a quantum circuit (unitary) that can explore the complete space it covers, the closer it resembles the Haar measure [31]. For an ansatz, as the expressibility increase, the circuit becomes more difficult to train due to flat regions in the cost function landscape. This raises the question, is it relevant/necessary to sample an ansatz with truly uniformly with the Haar measure?

Unitary t-designs are related to problems that use randomization or sampling from random unitaries. Unitary designs can be used as an alternative to sample the Haar measure. The unitaries in a unitary design are evenly spaced points across the unitary group and with a subset of the full group, an exact expression can be obtained, which originally would require measurement of the full set. This is shown in eq. (2.41):

$$\frac{1}{K}\sum_{k=1}^{K} P_{t,t}\left(U_k\right) = \int_{\mathcal{U}(d)} P_{t,t}(U) d\mu(U) \tag{2.41}$$

This equation is exact and holds for all $P_{t,t}$, where P is a polynomial of order t, K is the set of unitaries $\{U_k\}$ and $d\mu$ is the uniform Haar measure [63]. Simply said, this expressions shows that an exact average over a full group can be taken without having to sample the full space. So unitary t-designs are a simplified exact version of Haar random unitaries. This property can for example be very useful when measuring the average fidelity after a noisy quantum operation. A good example is shown by PennyLane and will now be demonstrated [63]. Consider the case where one applies a unitary V to an initial quantum state $|0\rangle$ via a noisy channel Λ. The fidelity of this operation can be expressed as:

$$F(\Lambda, V) = \left\langle 0 \left| V^{\dagger} \cdot \Lambda(|0\rangle\langle 0|) \cdot V \right| 0 \right\rangle = 1 \tag{2.42}$$

Which would equal $F = 1$, in the ideal case and $F < 1$, when the noise factor is non-zero. To get the average fidelity, one would have to Haar-random unitary U states in order to sample this state a large number of times and evaluate the following integral:

$$\bar{F}(\Lambda, V) = \int_{\mathcal{U}} d\mu(U) \left\langle 0 \left| U^{\dagger} V^{\dagger} \Lambda\left(U|0\rangle\langle 0|U^{\dagger}\right) V U \right| 0 \right\rangle \tag{2.43}$$

In the limiting case, one needs to sample this an infinite amount of times to get to the exact value. Unitary-t designs can bring a solution here. Since the elements in the above equation from a polynomial of degree 2, it matches the definition of a unitary 2-design. By by using the set of K unitaries that form the 2-design, one can compute the average fidelity with a finite number of states:

$$\frac{1}{K}\sum_{j=1}^{K} \left\langle 0 \left| U_j^{\dagger} V^{\dagger} \Lambda\left(U_j|0\rangle\langle 0|U_j^{\dagger}\right) V^{\dagger} U_j \right| 0 \right\rangle = \int_{\mathcal{U}} d\mu(U) \left\langle 0 \left| U^{\dagger} V^{\dagger} \Lambda\left(U|0\rangle\langle 0|U^{\dagger}\right) V U \right| 0 \right\rangle \tag{2.44}$$

But how does this affect barren plateaus? In an increasing number of papers such as [16, 31, 50, 78], barren plateaus have been related to unitary 2-designs. If an ansatz, with sufficient depth, forms a 2-design, then with high probability it will result in a barren plateau [50]. It can be concluded that unitary t-designs are nice properties to have when computing the average properties over a space, but not when optimizing a parameterized quantum circuit. Approximate 2-designs can be prepared by a polynomial depth random circuit [9, 29], indicating that polynomial depth circuits could also lead to barren plateaus.

2.9.2. Ansatz Expressibility

As the number of qubits in a quantum circuit increases, the Hilbert space grows exponentially. For a parameterized ansatz this means that searching for the correct solution becomes increasingly difficult as the search space increases. This is comparable to the curse of dimensionality, where as more dimensions are added to a problem, finding the solution often takes an exponentially increasing amount of time. As mentioned earlier in this report, the ansatz should have a suitable level of expressibility for the problem that it is trying to solve. As the Hilbert space increases in size it generally becomes more difficult to train the parameters if the ansatz has a high level of expressibility. In the work performed by Holmes et al. [31] it is studied whether there is a clear relation between the trainability of an ansatz and its expressibility. Here the expressibility of an ansatz is determined by the how uniformly it explores the unitary space. This study is performed for general ansatze, rather than the work in [16], where only barren plateaus for ansatze that form 2-design were considered. It is found that the more expressive an ansatz is, the smaller the variance in the cost function gradient (flat landscape). An example of this relation is shown in fig. 2.9. Here two different ansatze and their expressibility are shown. The ansatz with low expressibility might not include the solutions to some problems in its search space and can lead to both large and small gradients in the cost function landscape. For the highly expressible ansatz the chance that a solution lies in its search space is larger, but the cost function landscape is often much flatter.

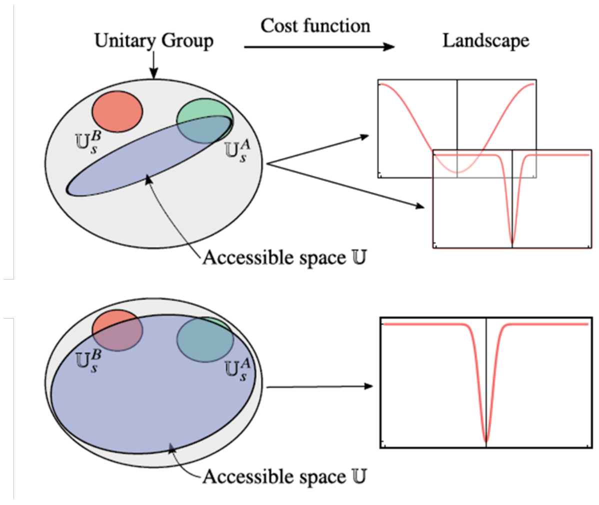


Figure 2.9: The difference between an ansatz with high and low expressibility. The unitary group indicates the complete search space, while the accessible space covers the region covered by the Ansatz. U_S^A and U_S^B are the solution to 2 arbitrary problems. Where the first ansatz is only able to express the solution to problem A and the second ansatz to both problems. The ansatz with a low expressibility can have both small and large gradients in the cost function landscape, while for an ansatz with high expressibility the cost functions are predominantly flat. [31]

While very expressive ansatze are more likely to exhibit barren plateaus, the opposite is not necessarily true. An ansatz which has a very low level of expressibility can still lead to the occurrence of barren plateaus. This result also corresponds to what is found in [16] and shown in fig. 2.10. This also means that it is not possible to add a lower bound to the gradient of an ansatz in terms of expressibility. Continuing with the work by Holmes et al. [31], they propose various methods of reducing the expressibility of an ansatz with the aim of improving trainability. These are correlating parameters, restricting rotation direction and restricting rotation angles of the parameterized quantum gates. Correlating the parameters is done by rotating specific qubits or circuit layers by the same angles. Restricting the rotational direction to only a single dimension at a time can help improve the scaling of the vanishing gradient in number of qubits by limiting the search space. For example, this is done by only applying rotations around the Y-axis instead of X, Y and Z at the same time. However, in a large number of ansatze only a single rotation direction is used, thus leaving this option not really adding much value for those problems. Finally restricting the rotational angles is also proposed as a method to limiting the expressibility, however this time it does not help trainability as it only limits the search space but is not related to the formation of the cost landscape. Thus training can still be difficult, while the chance of excluding the solution is also increased. The only time when restricting the rotation angles can be useful is when the problem is initialized close to the solution. This way the variance of the gradient is greatly improved, but requires knowing roughly where in space the solution lies, which for most problems is not feasible. [31]

2.9.3. Cost Function Influence

The vanishing gradient of the cost function is greatly influenced by the formulation of the cost function and as mentioned in section 2.7, local cost functions can help with the avoidance of barren plateaus. It is proven that barren plateaus can be avoided if the depth of the ansatz is limited to $\mathcal{O}(log(n))$. This way, systems with a local cost function could be trained up to 100 qubits, while with a global cost function training on problems larger than 20 qubits became unsuccessful [16]. An example of how the trainability of an ansatz progresses is shown in fig. 2.10. Here it is shown that as the ansatz becomes deeper the parameters become more difficult to optimize. For a global cost function all circuit depths can be difficult to optimize, while for local cost functions trainability reduces as the depth is increased. This is closely related to the expressibility of the ansatz that is used.

One might think that barren plateaus are also affected by the difficulty of the problem that one is trying to optimize, but even in very simple problems barren plateaus can occur. In the same work by Cerezo et al. [16], a simple example is shown where a tensor product ansatz of R_X gates is used in combination with the global cost function $C_G = 1 - \Pi_{j=1}^{n} cos^2(\theta^j/2)$. This is compared to a local cost

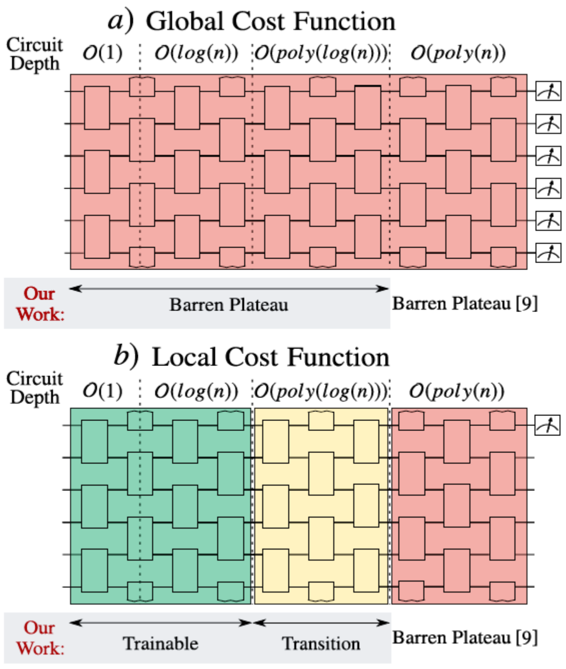


Figure 2.10: Simplified version of the effects of a global cost function in $a)$ and a local cost function in $b)$. [16]

function: $C_L = 1 - \frac{1}{n}\Sigma_{j=1}^{n} cos^2(\theta^j/2)$. The goal of this example problem is finding the angles θ^j such that $V(\theta)|0\rangle = |0\rangle$. It can be shown that for the global cost function the variance results in the left equation in eq. (2.45) which scales exponential in the number of qubits, while for the local cost functions the equation on the right is obtained which scales polynomially with n.

$$\mathrm{Var}\left[\frac{\partial C_{\mathrm{G}}}{\partial \theta^j}\right] = \frac{1}{8}\left(\frac{3}{8}\right)^{n-1}, \quad \mathrm{Var}\left[\frac{\partial C_{\mathrm{L}}}{\partial \theta^j}\right] = \frac{1}{8n^2} \tag{2.45}$$

In fig. 2.11 the landscape of the corresponding cost functions as mentioned above are shown. Here it also becomes clear that for the global cost function a steep well occurs with increasing problem size, while for the local cost functions larger gradients are maintained.

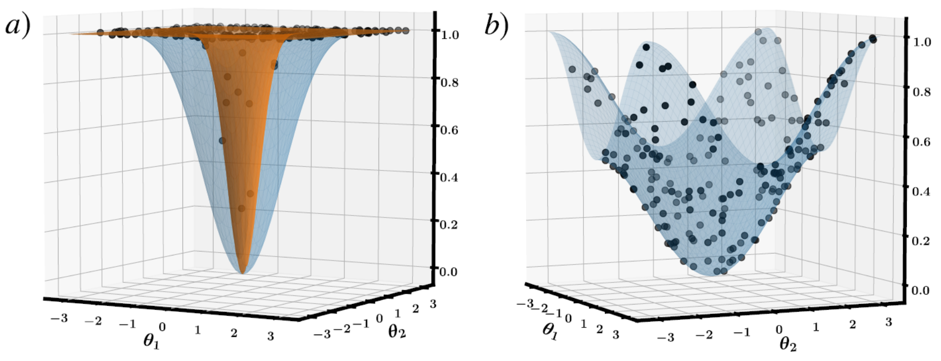


Figure 2.11: Cost function landscapes for a global cost function shown in $a)$ and local cost functions in $b)$. In $a)$, the light blue landscape is obtained for the problem containing 4 qubits, while the orange (narrow) well is obtained for the same problem with 24 qubits. In $b)$ the shape is independent of problem size. The black dots shown in both figures are 200 Haar distributed points.[16]

The work performed by Uvarov et al. [78] showed that for a general variational ansatz and Hamiltonian, a lower bound on the gradient of the cost function gradient can be found. They showed that the variance of the gradient is a weighted sum of the variances for the individual Pauli strings forming the cost function Hamiltonian and that the variance for an individual Pauli string can be bounded from below using the width of the causal cone of that string. Causal cone here means the number of qubits on which a unitary can act non-trivially. From this statement they conclude that barren plateaus do not only occur merely because of the cost function but are also affected by the ansatz structure. Uvarov et al. describe locality of a Pauli string as the number of non-identity Pauli matrices contained in a string. Their results imply that locality is not the only factor in the occurrence of barren plateaus but that the interaction between the ansatz and the cost function Hamiltonian is more important. These results demonstrate that gates on the edges of a quantum circuit are potentially less prone to barren plateaus than those in the middle, when in line connectivity is used. [78]

2.9.4. Noise Induced Barren Plateaus

Barren plateaus are not only introduced due to the cost function or ansatz, but can also be induced by hardware noise. These types of barren plateaus are called Noise Induced Barren Plateaus (NIBPs). Work performed by Wang et al. [81] investigated the effect of noise on variational quantum algorithms and concluded that noise can cause the training landscape to have flat spots and induce barren plateaus. They also concluded that any of the strategies to avoid noise-free barren plateaus do not appear to solve the noise induced barren plateaus. For 'standard' barren plateaus that have been discussed previously, a main aspect is that the variance of the gradient reduces exponentially in the number of qubits and that the global minimum of the cost function lies in a deep well. The significant difference with noise induced barren plateaus is that these flatten the complete landscape. Meaning that there is no deep well where the true solution lies. Avoiding noise induced barren plateaus is a challenging task, as layer-wise training, correlating ansatz parameters and artificially increasing gradients does not provide any way to resolve them. The main conclusion is that circuit noise and thus circuit depth should be kept to a minimum. When the noise introduced to the circuit is smaller, the cost function values and gradients can be computed more exact and the well will be better defined.

The original VQLS paper by Bravo-Prieto et al. [10] also briefly mentions the effect of hardware noise. While implementing the VQLS algorithm on a hardware device, noise resilience was found while training the ansatz. This resilience showed in the form of optimal parameter resilience, where the VQLS algorithm was still able to train the parameters while various noise sources were present. The work by Sharma et al. [73] also describes this phenomenon, indicating that noise resilience might help optimizing quantum circuits in the NISQ era.

2.9.5. Entanglement Induced Barren Plateaus

Another, perhaps more unexpected, phenomenon that leads to barren plateaus is that of entanglement. As a quantum system contains a higher level of entanglement between qubits, training of the ansatz parameters becomes more difficult. The work by Marrero et al. [58] showed that as the volume of the hidden units in a QNN is increased, the probability of finding barren plateaus increases. For Haar-random pure states and thermal states of random Hamiltonians the gradient of the cost function will vanish exponentially in the number of hidden units. The ansatz used in the VQLS algorithm can be regarded as a type of QNN, and the entanglement within this ansatz thus has to be limited in order to prevent barren plateaus. These results are in line with what was found with the results of ansatz expressibility and the occurrence of barren plateaus.

2.9.6. Shadow Protocol

Sack et al. [68] described the phenomenon of barren plateaus as: "The gradients of the cost function are on average zero and deviations vanish exponentially in system size.". Their study also introduced a new term of weak barren plateaus (WBPs), which emerge when the level of entanglement of a local subsystems exceeds a certain threshold by the entanglement of a mixed state. When WBPs are absent of a problem it is guaranteed that standard BPs also do not occur [68]. WBPs can be efficiently diagnosed using density matrices, and by doing this it can be efficiently shown whether a problem contains a barren plateau. If a unitary circuit in a VQA forms a 2-design, it implies emergence of a BP and thus a WBP. WBP are mathematically defined by points where the second Renyi entropy $S_2 = -\ln(tr\rho_A^2)$ of any subregion of k-qubits satisfies $S_2 \geq \alpha S^{Page}(k, N)$, where the Page entropy, as defined in eq. (2.46), in the limit $k \ll N$ corresponds to the maximal possible entanglement of subregion A. ρ_A is the reduced density matrix, where A is the subset of qubits which are measured, N is the number of qubits and k the locality of the Hamiltonian. [68]

$$S^{\text{Page}}(k, N) \simeq k \ln 2 - \frac{1}{2^{N-2k+1}}, \tag{2.46}$$

The learning rate (step size) used in the classical optimizer also influences how likely it is for BPs to occur for a VQE algorithm. These findings should also readily extend to other variational algorithms. One of the proposed solutions is the 'shadow protocol', which employs a classical shadow estimation and tracks the level of entanglement in subregions of the problem. Shadow estimation or shadow tomography is a method to directly estimate properties of an unknown state without performing full state tomography [68]. This way properties such as subsystem purities or gradients can be extracted. However, for this method to work an N-qubit classical shadow has to be constructed, which takes N measurements for each qubit and has to be repeated T amount of times, to then average over all samples to obtain an estimation of $\rho(\theta)$. Only for $T \to \infty$ does this method become exact, but study has shown that optimization happens more rapid. Even though this method might help avoiding barren plateaus, the large number of additional measurements make the implementation on NISQ devices questionable.

Study on QNNs has shown that any circuit of sufficient depth that forms a 2-design will produce an ansatz state on a barren plateau in the quantum landscape [50]. Classical neural networks can optimize up to billions of neurons, so why is this more difficult for QNNs? This is due to two differences. The first is that classically the gradient of a neural network vanishes in the number of layers, while for the quantum case it happens in the number of qubits. The second reason is that the complexity of computing the expectation value is larger in the quantum case compared to the classical one. Computation of the gradient for a classical neural network scales with $\mathcal{O}(log(1/\epsilon))$, while for a QNN it scales with complexity $\mathcal{O}(1/\epsilon^{\alpha})$, here α is a term depending on the eigenvalues of the evaluated unitary [38, 50]. As a result, standard gradient based optimization is challenging to perform in polynomial time on a quantum device.

2.9.7. Error Mitigation

On a final note regarding barren plateaus, error mitigation will be discussed. The aim of error mitigation is that the impact of noise is tried to be minimized. As discussed previously, noise can lead to the occurrence of NIBPs and minimizing it could help improving trainability of algorithms. This has been studied by Wang et al. [80]. Their work investigated whether error mitigation techniques can help with the trainability of noisy algorithms and showed a few different results. First, if a VQA is suffering from exponential cost concentration then a wide range of error mitigation methods do not work. Examples of methods they mention are: Zero Noise Extrapolation, Virtual Distillation, Probabilistic Error Cancellation and Clifford Data Regression. This means that the required resources, such as the number of shots or circuits, remains exponential when finding the solution to a problem. Other results showed that error mitigation methods can, depending on the case, both improve and worsen the trainability of a VQA. Finally, they also noted that using post-processing method for error mitigation might be less efficient for improving trainability, however that by using sufficient resources NIBPs can be avoided. Thus by using more advanced error mitigation techniques, that operate during execution of a quantum circuit rather than when all the data has been gathered, a more effective method to noise reduction could be obtained [80].

2.10. Classical Optimizers

Optimizers are theorized to play a role in the avoidance of barren plateaus. Due to the different nature of various optimizers, such as gradient-based and gradient-free optimizers, some of them are more susceptible to barren plateaus than others. Since barren plateaus are defined as areas in the cost function landscape where the gradient is vanishing exponentially in the number of qubits, an immediate conclusion might be that it only affects gradient based optimizers. However, even gradient free optimizers do not rule out the problem of barren plateaus [6]. Gradient based optimizers often take the form:

$$\theta^{t+1} = \theta^{t} - \eta\nabla_{\theta}C(\theta) \tag{2.47}$$

Here θ^t are the ansatz parameters for iteration t and η is the step size parameter used to scale the gradient of the cost function $\nabla_\theta C(\theta)$. So if the term $\nabla_\theta C(\theta)$ is a value close to zero, the second term on the right hand side will become very small. What might seem as a simple solution to this problem, is increasing the step size parameter η in order to increase the effect of this second term. However, this is not a suitable option as it increases the risk of overshooting the area where the global minimum is located. Since the problem of barren plateaus often results in a higher-dimensional cost function landscape where the solution lies in a shallow deep well, other methods have to be resorted to.

The work performed by Pellow-Jarman et al. [61] studied the performance of a wide range of classical optimizers on a VQLS algorithm. They did this for problem size of $n = 3, 4, 5$ qubits, with the goal of solving $A|x\rangle = |b\rangle$. Three different matrix problems in the form of $A = \sum_i c_i P_i^j$ were used. Here c_i is an arbitrary coefficient and P can take form of the identity matrix or one of the Pauli matrices. For all problems the right hand side took the form of $|b\rangle = H^{\otimes n}|0\rangle^n$. One of the advantages of using this state is that it is easily implementable on a real hardware device. In order to prevent influences of the parameter initialization, the same 100 sets of input parameters are used for each optimizer. This way it is ruled out that one optimizer is performing better due to 'luck' of getting initial parameters closer to the solution. The same local cost functions as described by Bravo-Prieto et al. [10] are used. Pellow-Jarman et al. evaluated the performance of an optimizer by looking at the average rate of convergence, the average termination cost values and the influence of noise. They compared results for exact statevector simulations, simulations including shot noise, and simulations including both shot noise and hardware noise.

For the statevector simulations all optimizers performed similarly, with no significant differences between gradient-based and gradient-free optimizers. Once shot noise is introduced, the performance of gradient-based optimizers generally outperforms that of the gradient-free optimizers. An exception to this is the gradient-free SPSA (Simultaneous Perturbation Stochastic Approximation) optimizer, which only relies on an approximation of the gradient. When a realistic noise model is added to the simulation, all optimizers perform significantly worse. It may become increasingly difficult for gradient-based optimizers to accurately compute the gradient when noise is present, as evaluating analytic gradients requires many circuit evaluations. Again, SPSA seems to be the best performing algorithm when including both shot noise and hardware noise. [61]

In short, from this study it can be concluded that without hardware and shot noise, the optimizers perform similarly. When shot noise and hardware noise are included all optimizers perform significantly worse, and again there is no clear distinction between the performance of gradient-based and gradient-free optimizers. SPSA performed slightly better than the other optimizers under noise, but this could be depending on the hyperparameters of the problem, which were kept constant in the mentioned work.

The optimizers used in the work mentioned above are all readily implemented state-of-the-art optimizers, which are originally designed for classical algorithms. An important aspect of quantum algorithms is measurement of the output. For this a certain number of shots is used, which after post-processing influences the accuracy of the output. Kübler et al. [39] proposed an algorithm specifically designed to take into account the shot number. The individual Coupled Adaptive Number of Shots (iCANS) algorithm is an adaptive algorithm that changes in two different aspects: changing the number of shots depending on the iteration and on the partial derivative that is given. Two different versions of the iCANS algorithm are proposed, one with and one without an adaptive learning rate. Results show that this method can improve convergence over the whole range of iterations compared to other optimizers.

This algorithm might give speedup over existing algorithms by reducing the number of shots when the accuracy is of less importance. But it does not seem to help with problems where convergence is already difficult, as the classical optimizers could also increase the number of shots by a fixed amount. Furthermore, this optimizer has no use in a wavefunction simulator, as the results there are exact and no shots are used and will be of limited use for the proposed work. On a final note, an interesting result is that in this study SPSA seems to work better than the ADAM optimizer, which agrees with what has been found in [61].

2.11. Parameter-Shift Rule

Gradient evaluation is a very important part in a lot of optimizers, and especially with the phenomenon of barren plateaus, accurate gradient information is of great use. Luckily there exists a method to obtain exact values for the gradient of the cost function with respect to an input parameter on a quantum computer. This can be done with the Parameter-Shift Rule (PSR). The PSR has similarities to a finite difference method, but the main differences are that the step size is not bound to be small and that the obtained value of the derivative is exact rather than an approximation [42, 49, 51, 71]. In the work by Mitarai et al. [51], the PSR is shown as:

$$\frac{\partial \langle B \rangle}{\partial \theta_i} = \frac{\langle B \rangle_j^+ - \langle B \rangle_j^-}{2} \tag{2.48}$$

Here $\langle B \rangle$ means the expectation value of a certain observable, the plus and minus sign refer to the positive and negative shifting of the j-th angle θ of the rotation gate. This is one of the first formulations of the parameter-shift rule. Apart from the fact that B should be an observable, it does not put any limitations on the nature of this observable.

To make the comparison between the central finite difference method and the PSR, eq. (2.49) and eq. (2.50) are used. In these equations f is the function which evaluates the cost, θ is the parameter used for initialization of the ansatz. In this case θ is shown as a single parameter, but this can be extended to more parameters. However, the PSR is only valid for shifting a single parameter at a time. s is the step size, or shift size of the parameter. For eq. (2.49), the finite difference method, s must be sufficiently small, otherwise the approximation of the gradient is not accurate. The form of the PSR as shown in eq. (2.50), is known as the standard form and here a value of $s = \pi/2$ must be used. Here it immediately becomes clear that the shift for PSR can be made much larger than what is possible for classical finite difference methods.

$$\frac{\partial}{\partial \theta} f(\theta) = \frac{f(\theta + s) - f(\theta - s)}{2s} \tag{2.49}$$

$$\frac{\partial}{\partial \theta} f(\theta) = \frac{f(\theta + s) - f(\theta - s)}{2} \tag{2.50}$$

The form shown in eq. (2.50) is a limited form of the PSR and only valid for a shift of $\pi/2$. In some instances it might be desirable to shift the parameter by a different angle, due to for example hardware related arguments. For this the continuous form of the PSR as shown in eq. (2.51) can be used. Here s is free to be chosen at any angle, apart form multiples of π due to the vanishing of the $sin(s)$ term.

$$\frac{\partial}{\partial \theta} f(\theta) = \frac{f(\theta + s) - f(\theta - s)}{2sin(s)} \tag{2.51}$$

The evaluation of the PSR can become computationally expensive quite fast. For each parameter during (gradient-based) optimization a derivative has to be obtained. Since the gradient is obtained by evaluating at 2 locations this scales with $2m$ measurements, times the number of shots needed for the desired accuracy. This means that as the number of parameters becomes larger and larger, the total time taken for construction of all the gradients will increase drastically.

3 Variational Quantum Linear Solver

This section covers the different components needed for the implementation of the VQLS algorithm. First the base of the circuit, the ansatz is explained, after which the generation of the right hand side vector is covered. Then the decomposition which is used and the cost functions are layed out. After which the post-processing procedure for the evaluation of the raising and lowering operators on a quantum machine is explained. Then the parameter shift rule, which is an exact method for the evaluation of cost function gradients with respect to the ansatz parameters is discussed. At last, the classical optimizer for the algorithm and the approach of rescaling the quantum solution to match the classical non-scaled solution are elaborated.

3.1. Ansatz

Two different types of ansatze are used for the VQLS algorithm during this work. These are the basic HEA with only R_y-rotations and CNOT ladders entangling all qubits. This ansatz is defined as eq. (3.1), where alternating blocks of rotational gates and entangling gates are applied. Here the p-th term refers to the p-th ansatz layer, $U_{rotation}$ and $U_{entangling}$ are the unitary layers used for the rotations and entanglement and θ^p are the parameters used for the rotational gates in the p-th layer.

$$U(\theta) = \prod_p U^p_{rotation}(\theta^p)U^p_{entangling} \tag{3.1}$$

The rotations are performed by only using R_y gates, such that the amplitudes of the quantum states remain real valued. Mathematically the operations can be written as $U_{rotation} = \sum_l R_y(\theta_l)$ and $U_{entanglement} = \sum_{l=0}^{n-2}\sum_{m=l+1}^{n-1} CNOT(l,m)$. Where θ_l corresponds to the rotations applied to the R_y gates in layer l. These rotations are different for each individual rotation gate. In fig. 3.1 a 3-qubit example of this ansatz is shown. Here two layers of the ansatz are used. After each layer a vertical barrier is shown to better indicate the individual layers.

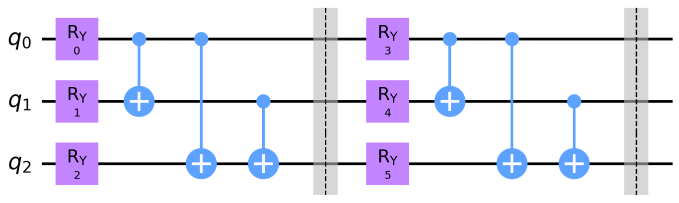


Figure 3.1: HEA shown for 3 qubits and 2 layers.

The second ansatz that is used, is the QAOA ansatz as shown in the paper by Liu et al. [44]. This ansatz was as already introduced in section 2.6, but will now be explained in more detail. The ansatz consists of several parts. First a layer of Hadamard gates is used to generate an equal superposition between al possible states. Then the driver and mixer Hamiltonians are used to complete the ansatz.

In fig. 3.2 a 3-qubit example of a single layer is given. The driver Hamiltonian is the part between the Hadamard gates and the first vertical barrier, while the mixer Hamiltonian consists of the 3 R_x gates between the last two barriers.

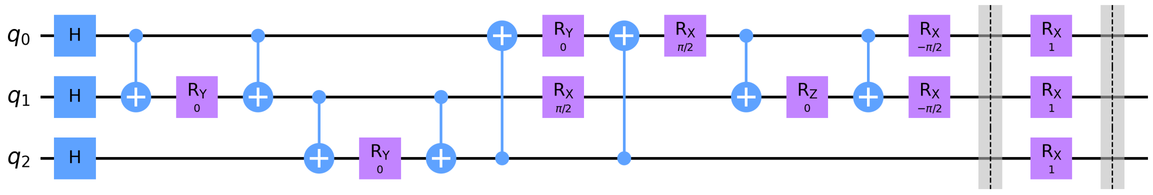


Figure 3.2: QAOA shown for 3 qubits and 1 layer.

In fig. 3.3 and fig. 3.4, different circuit parameters are shown for the single layered HEA and QAOA. The 5 parts in the legend are the ansatz depth, which is just the circuit depth of the ansatz. 'Depth Controlled Ansatz' refers to the the controlled ansatz, as how it is used in the VQLS algorithm. When a controlled-unitary is applied to a circuit the total number of gates increases over the standard non-controlled unitary, which is why it is an important measure to show. The non-local gates is the number of multi-qubit gates, such as the CNOT gates. The final two parameters are the gate count of the ansatz and the gate count of the controlled ansatz. For all values of the controlled ansatz, it is obtained after the controlled-unitary is decomposed into gates by Qiskit. From the figure it becomes clear that the HEA in this fashion scales exponentially in the number of qubits. Which would not make it a suitable ansatz for very large problems. However, since its expressibility is large due to the entanglement it will be used as a test case for the smaller problems covered in this work.

An advantage of the QAOA ansatz is that only local entanglements are used between different qubits, as opposed to the HEA shown above where each qubit is entangled with every other qubit. This means that it would be easier to implement on a hardware device. A downside of this method is that the circuit depth is relatively large. Even though it is large, it does scale linearly in system size as is shown in fig. 3.4. When comparing the numbers to that of the HEA on the left figure, it becomes clear that both the number of gates and circuit depth of the QAOA is larger for the problem sizes at hand.

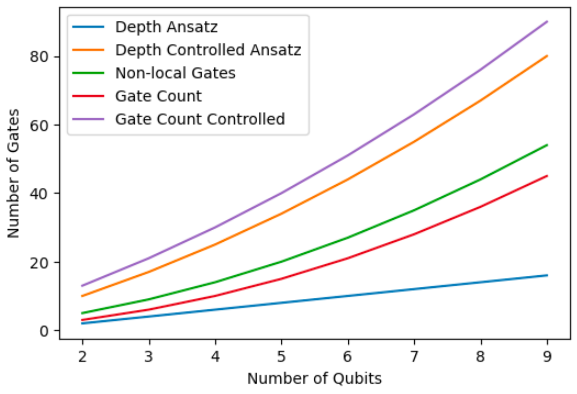


Figure 3.3: Scaling of the HEA circuit depth versus the number of qubits.

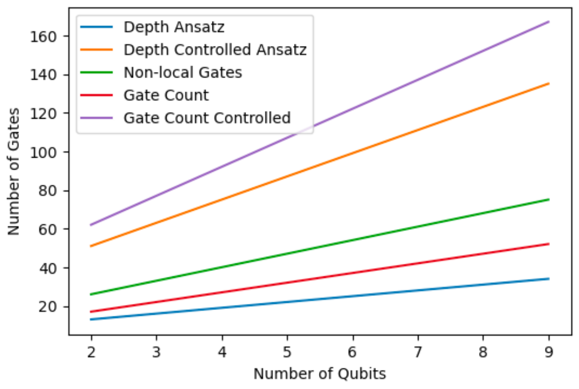


Figure 3.4: Scaling of the QAOA circuit depth versus the number of qubits.

3.2. Generation of the Right-Hand Side Vector

For the problem at hand the right-hand side vector $f(x) = x$ is used on the domain $\phi \in [0, 1]$. This is a continuous function, which is discretized in order to be shaped into a vector. The discretization is done by mapping the continuous vector onto a set of evenly spaced nodes. The obtained vector is then normalized in order for it to be converted to a quantum state. This is where another problem enters. The generation of an arbitrary quantum state is, in some cases, a non-trivial problem. In the case where the right-hand side vector is an evenly distributed state such as $(1/N^2, ..., 1/N^2)$, the implementation on a quantum computer is straight forward. By applying the Hadamard-gate on each qubit, a quantum state with equally distributed amplitudes is immediately constructed. However, for the case where $b = x$, this is not so simple. In order to construct an arbitrary quantum state another ansatz is used, such that $U(\theta)|0\rangle = |b\rangle$. But finding the correct angles θ can again be a very difficult process. Ideally one would use a measure of the fidelity as a comparison of how close the generated state $|b\rangle$ is to the true value. But in order to measure $|\langle\psi|b\rangle|$ on a quantum computer, the vector $|b\rangle$ already must be prepared, which immediately returns us to the original problem. Since Qiskit performs a complete statevector simulation, it luckily is possible to retrieve the full quantum state and exactly compute $|\langle\psi|b\rangle|$, such that the optimization for $U(\theta)|0\rangle = |b\rangle$ can be done. However, in a real world scenario this is of course not possible and a different solution must be resorted to.

Another simple method in Qiskit, which allows for the generation of a normalized quantum state, is that of the StatePreparation method. This allows for the initialization of an arbitrary quantum state on a set of qubits. This unitary can then also be applied in a controlled way, as is required in some part of this implementation of the VQLS algorithm. For simple implementations of the algorithm this approach will be used, while for the complete quantum implementation the previously mentioned method of $U\theta)|0\rangle = |b\rangle$ will be used.

3.3. Matrix Decomposition

The decomposition of the A and A^2 matrices as introduced by Liu et al. [44] is used as mentioned in section 2.8. This decomposition is an important part of the algorithm but easy to implement. The decomposition is done by using the identity operator I, the raising operator σ_+ and the lowering operator σ_-. This way a decomposition is obtained, in which the number of terms scales linearly as the problem size is increased. One difficulty over that of simply using a Pauli decomposition, is that the raising and lowering operator are not directly measurable on individual qubits. In order to be able to measure the raising and lowering operators, the full quantum state is rotated into the Bell basis. Meaning that the observables effectively become global observables. This has big consequences for the cost functions which can be used and the post-processing which is required to evaluate the expectation values. Both are discussed in the following two sections.

3.4. Cost Function

An important part of the algorithm is the type of cost function which is used. In section 2.7.1 some examples of cost functions have been shown. The two main ones discussed in this report both rely on the effective Hamiltonian approach. These are the standard cost function as shown in eq. (3.2) and the normalized cost function as in eq. (3.3).

$$C_{standard} = \langle\psi|A^2|\psi\rangle - |\langle b|A|\psi\rangle|^2 \tag{3.2}$$

$$C_{normalized} = 1 - \frac{|\langle b|A|\psi\rangle|^2}{\langle\psi|A^2|\psi\rangle} \tag{3.3}$$

In these cost functions $|b\rangle$ is the normalized right hand side vector, $|\psi\rangle$ is generated from the parameterized ansatz and A is the discretized Poisson equation.

In section 2.7.1 local cost functions were introduced. These cost functions make use of local properties of the qubits and in this way help reducing the problem of barren plateaus. In the case where, for example a Pauli decomposition is used where one is able to directly measure individual qubits, a local cost function indeed is possible. But since the decomposition when using Pauli gates results in an exponentially increasing number of terms in the problem size, it is an inefficient approach. Ideally a different decomposition which scales efficiently in the problem size must be used. However, since the decomposition used for the A and A^2 matrix relies on the usage of global operators, it is not possible to use local cost functions. Thus, the global standard and normalized cost functions are used in this work.

3.5. Post-Processing of Measurements in the Bell Basis

This section covers the post-processing of the measurements in the Bell basis. First the process of generating the correct quantum states is explained, after which the exact post-processing of the measurement data is covered.

3.5.1. Generation of the Quantum States

The terms used in the decomposition introduced by Liu et al. [44] consist of raising and lowering operators. These operators are defined as $\sigma_+ = |0\rangle\langle 1|$ and $\sigma_- = |1\rangle\langle 0|$, which are neither Hermitian, nor normal operators. Hermitian operators ($A^\dagger = A$) are often most easily measured since these are readily implemented on a quantum computer. However, these are only a subset of the complete set of all measurable operators, which are the normal operators ($AA^\dagger = A^\dagger A$) [32]. The authors of [44] found a way to transform the raising and lowering operators into measurable normal operators. In eq. (3.4) and eq. (3.5) the observables to measure $\langle\psi|\sigma_\pm|\psi\rangle$ are shown.

$$\begin{bmatrix} 0 & \sigma_+ \\ \sigma_+^\dagger & 0 \end{bmatrix} = |\varphi_{11}^+\rangle\langle\varphi_{11}^+| - |\varphi_{11}^-\rangle\langle\varphi_{11}^-| \equiv O_{11} \tag{3.4}$$

$$\begin{bmatrix} 0 & \sigma_- \\ \sigma_-^\dagger & 0 \end{bmatrix} = |\varphi_{12}^+\rangle\langle\varphi_{12}^+| - |\varphi_{12}^-\rangle\langle\varphi_{12}^-| \equiv O_{12} \tag{3.5}$$

Here $|\varphi_{ij}^\pm\rangle$ represent Bell states such as $|\varphi_{11}^\pm\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ or $|\varphi_{12}^\pm\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle)$. These states are measurable by rotating them from the Bell basis into the computational basis, which is done by means of an ancilla qubit, Hadamard gates and CNOT gates. In eq. (3.4) and eq. (3.5), the operators have a size twice as large as what the original operators had. This means that in order to evaluate such states on a quantum computer, the addition of an ancilla qubit is required. This extra qubit doubles the size of the quantum state, meaning that it correspond again to the size of the Hermitian form of the operators.

Normally, Bell states are generated by using a Hadamard gate on the first qubit followed by a CNOT gate with the control on the first qubit as is shown in fig. 3.5. For larger states the same principle is used. Even though only the two qubit states are, strictly speaking, called Bell states, in this report multiple qubit forms will also be referred to as such.

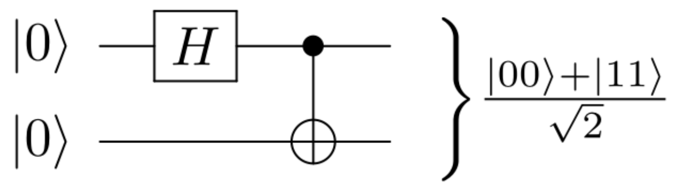


Figure 3.5: Circuit that generates a Bell state. [84]

The circuit shown above is used to rotate a quantum state into the Bell state, however in order to measure in the Bell basis a reverse type of this circuit is required. Since the original type of measurement is already in the Bell basis, it has to be rotated into the computational basis, which is done by the reverse of what is shown in fig. 3.5. Thus first, the CNOT gate(s) must be applied, followed by the Hadamard gate on the first qubit. As mentioned in section 2.8, the decomposition by Liu et al., makes use of the effective Hamiltonian approach where A and A^2 are discretized separately, and eq. (3.2) or eq. (3.3) is used to evaluate the cost. The expectation values of the tensor products of raising and lowering operators can then be computed by eq. (3.6) and eq. (3.7). Here O_{11} is used as an example observable.

$$\langle\psi(\theta)|\sigma_+|\psi(\theta)\rangle = \left\langle 0,1 \left| \langle\psi(\theta)|O_{11}|0,1\rangle \right| \psi(\theta)\right\rangle - i\left\langle 0,i1 \left| \langle\psi(\theta)|O_{11}|0,i1\rangle \right| \psi(\theta)\right\rangle \tag{3.6}$$

$$\left|\langle b|\sigma_+|\psi(\theta)\rangle\right|^2 = [\langle b,\psi(\theta)|O_{11}|b,\psi(\theta)\rangle]^2 + [\langle b,i\psi(\theta)|O_{11}|b,i\psi(\theta)\rangle]^2 \tag{3.7}$$

In the case where the quantum state prepared by the ansatz only contains real valued amplitudes, the second term on the right hand side in both equations can be left out. The $|0,1\rangle$ and $|0,i1\rangle$ states are represented by eq. (3.8) and eq. (3.9):

$$|0,1\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle), \quad |0,i1\rangle = \frac{1}{\sqrt{2}}(|0\rangle + i|1\rangle) \tag{3.8}$$

$$|b,\psi(\theta)\rangle = \frac{1}{\sqrt{2}}(|0\rangle|b\rangle + |1\rangle|\psi(\theta)\rangle), \quad |b,i\psi(\theta)\rangle = \frac{1}{\sqrt{2}}(|0\rangle|b\rangle + i|1\rangle|\psi(\theta)\rangle) \tag{3.9}$$

and are prepared by the addition of an ancilla qubit. For the latter, a controlled unitary for the $|b\rangle$ state and a negative controlled unitary for the $|\psi\rangle$ state is required. A 2-qubit problem (+ancilla qubit) example for these circuits is shown in fig. 3.6 and fig. 3.7. Here the purple block with $|\psi\rangle$ is a controlled unitary containing the ansatz and the block with $|b\rangle$ is a controlled unitary for the operation $U|0\rangle = |b\rangle$.

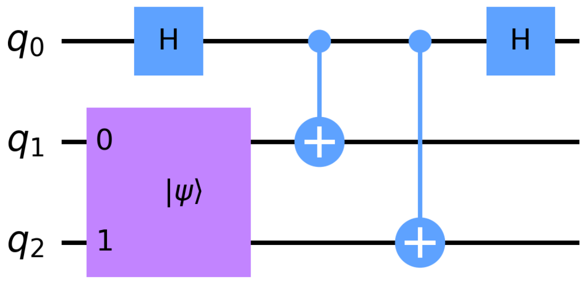


Figure 3.6: Circuit used to evaluate expectation values in the form of $\langle\psi| \otimes_j \sigma^j_\pm |\psi\rangle$.

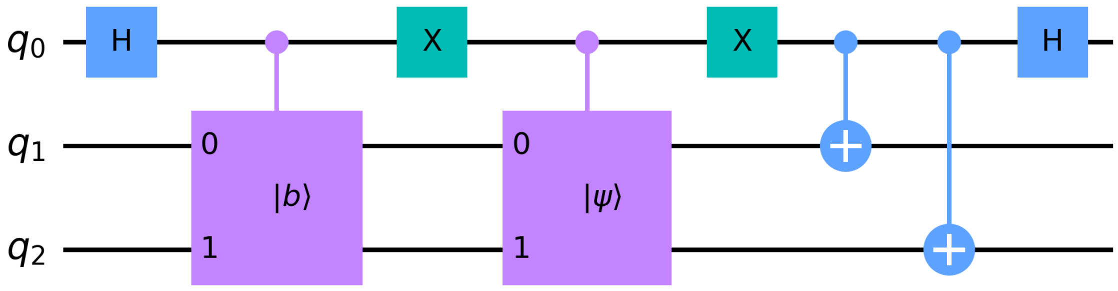


Figure 3.7: Circuit used to evaluate expectation values in the form of $\langle b| \otimes_j \sigma^j_\pm |\psi\rangle$.

As mentioned earlier, in the case of a Pauli decomposition, one is able to directly measure a certain operator by rotating it into the respective basis. However, with the Bell basis, the full state must be rotated rather than individual qubits. This has the consequence that, for the 2-qubit case, any operator of the 4 possible combinations $\sigma_\pm \otimes \sigma_\pm$ would require an identical circuit to evaluate the expectation values. These states can only be distinguished by post-processing of the measured data from the circuit.

3.5.2. Evaluation of Decomposed terms

In section 2.8 the method of decomposition of the A and A^2 matrices as by [44] was described. As mentioned above, a single circuit can be used to evaluate different operator tensor products, based on which post-processing of the bit strings is applied. The circuits used to evaluate the terms $\langle\psi|A_i^2|\psi\rangle$ and $\langle b|A_i|\psi\rangle$ in the cost function are shown in fig. 3.6 and fig. 3.7 respectively. When evaluating different operators with these circuits, it becomes clear that the first part of the circuit stays unchanged and that only the number of CNOT-gates at the end of the circuit is changed. An example will be given now. The two-qubit decomposition of the A-matrix, is as follows $A_2 = 2II - I\sigma_+ - I\sigma_- - \sigma_-\sigma_+ - \sigma_+\sigma_-$, here the symbol for the tensor product $\otimes$ is left out to simplify notation. The exact circuit which is needed to evaluate a certain tensor product of operators, is only determined by whether an I or $\sigma_\pm$ operator is used for each respective position in the decomposition. In the case of a $\sigma_\pm$ operator in the tensor product, a CNOT gate must be applied between the ancilla and the respective qubit. If an identity operator is applied no additional gates are needed. This means that for the II operator, no CNOT gates are needed in fig. 3.6. In the case of $I\sigma_+$ and $I\sigma_-$ the ancilla and second qubit must be entangled and for the $\sigma_-\sigma_+$ and $\sigma_+\sigma_-$ operators both the first and second qubit must be entangled with the ancilla by means of a CNOT-gate. The next step is then processing the correct data after repeated measurements.

Still using the 2-qubit problem example, in total 3 different qubits are measured (including the ancilla). This means that $2^3 = 8$ total bit string combinations are possible, which correspond to 4 different operators for an individual circuit. By adding and subtracting the relative probabilities of each bit string, the expectation value of the desired operator is obtained. For example, in order to evaluate $\langle\psi|\sigma_+ \otimes \sigma_-|\psi\rangle$, the following probabilities are used:

$$\langle\psi|\sigma_+ \otimes \sigma_-|\psi\rangle = p(100) - p(101) \tag{3.10}$$

Here $p(100)$ is the probability to obtain the quantum state $|100\rangle$ and $p(101)$ the probability to obtain $|101\rangle$. This might seem as a random combination of bit strings at first, however there is a simple structure behind it. Qiskit uses reverse ordering when assigning the bits. So in the circuits the top qubit (ancilla) is the last bit in a bit string as '000'. In the tensor products of operators a σ_+ operator fixes the respective bit to 1, the σ_- operator to 0 and for an identity operator both 0 and 1 must be taken into account. Furthermore, for each combination both the 0 and 1 state of the ancilla must be used, where the 0 state is added and the 1 state subtracted. An example of how this is done is given below:

$$\sigma_+\sigma_- \xrightarrow{\text{Translate to bits}} p(10) \xrightarrow{\text{Add ancilla}} p(100) - p(101) \tag{3.11}$$

Another example of a term in the 2-qubit case is $\langle\psi|I \otimes \sigma_+|\psi\rangle$. This time more terms are required due to the identity operator:

$$I \otimes \sigma_+ \xrightarrow{\text{Translate to bits}} p(01) + p(11) \xrightarrow{\text{Add ancilla}} p(010) - p(011) + p(110) - p(111) \tag{3.12}$$

The same process is used for all the terms in the decomposition of $\langle\psi|A_i^2|\psi\rangle$ and $\langle b|A_i|\psi\rangle$. The A^2 matrix is decomposed as $A^2 = B - C$. Here C is a zero matrix with only 1's on the first and last entry. This corresponds to measuring the full zero and one state of the respective qubits. The way this is computed is by $\langle\psi|C|\psi\rangle = p(n * 0 + 0) - p(n * 1 + 0)$, where n is the number of qubits. For the two-qubit case this results in $\langle\psi|C|\psi\rangle = p(000) - p(110)$. Regarding the B matrix, it is defined as $B = \otimes_j \sigma_\pm^j$ such that A^2 is completed when C is subtracted. The method used in the examples of $\sigma_+ \otimes \sigma_-$ and $I \otimes \sigma_+$ is also used for all other $\langle\psi| \otimes_j \sigma_\pm^j|\psi\rangle$ terms, as well for all terms in the decomposition of $\langle b| \otimes_j \sigma_\pm^j|\psi\rangle$.

With the scaling of this method it should be noted that for some of the terms, the amount of additions and subtractions of the probabilities scales exponentially in the number of qubits. Evaluation of the operator $I\sigma_+$ requires 4 probabilities, however for $II\sigma_+$ it requires 8. For each additional qubit the amount of post-processing terms doubles, meaning that there is an exponential scaling involved. However, it might be the case that for very large problems the number of counts used might be the limiting factor. Meaning that the total number of occurring probabilities stays limited and in practice not an exponential amount of terms have to be added, since most of them are negligible.

3.6. Parameter-Shift Rule

In order to get an exact value for the gradients of the cost function, the Parameter-Shift Rule is used. This topic is already explained in section 2.11. For the standard cost function in the form of $C = \langle\psi|A^2|\psi\rangle - |\langle b|A|\psi\rangle|^2$, the implementation of the PSR is straight forward. However, for the normalized cost function an extra step is needed.

3.6.1. PSR for the Normalized Cost Function

This extra step is required due to the fact that both terms in the cost function can be seen as individual functions, which depend on θ. Derivatives of multiple functions are additive, however not multiplicative. In fig. 3.8, for various terms, the angle shift of the first HEA parameter is plotted versus the cost function value. In the figure on the left it can be seen that $\langle\psi|A^2|\psi\rangle$, $|\langle b|A|\psi\rangle|^2$ and their subtracted value represent sine waves. On the right figure, the value for the normalized cost function is shown and it becomes clear that this does not represent a sine wave anymore, which is a crucial aspect for the PSR rule to work. This again shows that the ratio in the normalized cost function requires more attention when evaluating the derivative.

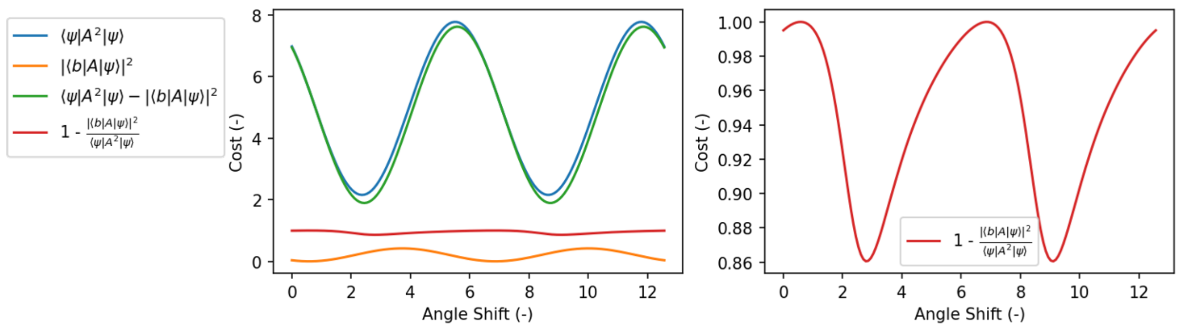


Figure 3.8: Cost function value versus a shift in rotation angle for the first HEA parameter. On the left the curves for the individual terms and the standard and normalized cost function are shown. On the right the normalized cost function is shown again on its own, in order to better see its shape.

The way this is solved is by using the quotient rule. If one defines the normalized cost function as $C_{norm} = 1 - f(\theta)$, where $f(\theta)$ is given by eq. (3.13), with $g(\theta) = |\langle b|A|\psi\rangle|^2$ and $h(\theta) = \langle\psi|A^2|\psi\rangle$.

$$f(\theta) = \frac{g(\theta)}{h(\theta)} \tag{3.13}$$

The quotient rule for evaluation of the derivative of $f(\theta)$ then reads:

$$f'(\theta) = \frac{g'(\theta)h(\theta) - g(\theta)h'(\theta)}{h(\theta)^2} \tag{3.14}$$

To reconstruct the full derivative vector for all the ansatz parameters eq. (3.14) must be evaluated separately for each parameter. The terms $g'(\theta)$ and $h'(\theta)$ are evaluated by the standard PSR as mentioned in section 2.11. This means that doing the quotient rule only leads to the evaluation of two extra expectation values, namely $h(\theta)$ and $g(\theta)$. These values can be re-used for all different parameters, as the shifting of the parameters only occurs during the evaluation of $g'(\theta)$ and $h'(\theta)$. So the total computational cost of implementing the quotient rule over the standard PSR is hardly increased.

3.6.2. PSR for the QAOA

Two assumptions for the PSR to be valid are that the operators must be Hermitian and that each rotational gate must be individually addressable. For the HEA this is indeed the case, however for the QAOA it is not. In this second ansatz the same parameter is used to rotate all the R_y gates in its respective layer. As a result the PSR cannot be used for this instance of the QAOA. The PSR works due to the fact that the rotation gates are cyclic. Changing the initialization of a gate by 2π results in the same effective angle. When rotating a certain gate and keeping track of the corresponding cost value, a sine wave forms. Due to this effect the PSR works, as is shown by the continuous form in eq. (2.51). In fig. 3.9 this effect is tested for both HEA and QAOA. Here a 2 qubit case is used. In both ansatze the first parameter is changed from $-\pi$ to π and the cost function value is tracked. For the HEA the curve is shaped as a sine wave. For the QAOA ansatz, this is not the case and thus the PSR does not work to accurately determine the gradient. Due to the fact that the shape is still periodic and the curve looks similar to what was found for the normalized cost function, it might be the case that an alternative to the standard PSR might provide a solution. However, this is not investigated in the current work

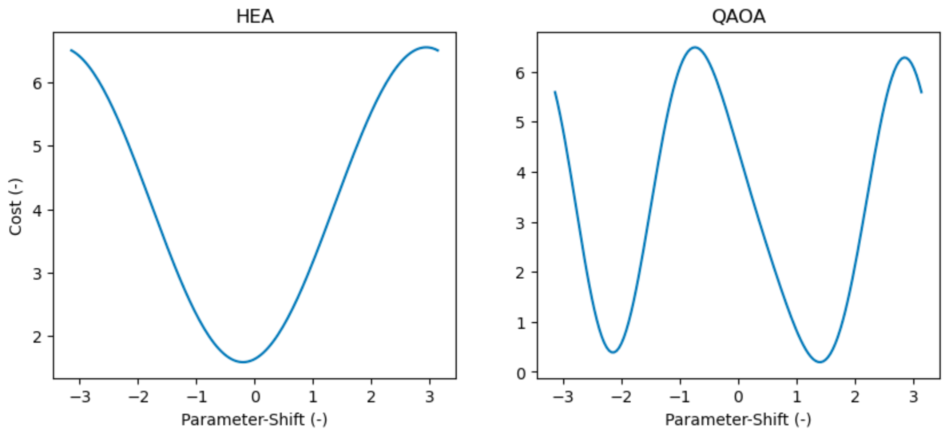


Figure 3.9: Cost function value versus a shift in the first parameter for both HEA and QAOA.

3.7. Classical Optimizer

The final part of the VQLS algorithm, is the classical optimizer. As mentioned in section 2.10 there are two main types of optimizers, namely gradient-based and gradient-free. When an exact implementation of the VQLS algorithm is used, it does not matter which type is used and any can be selected. When shot noise is taken into account, the gradient-based methods must use the PSR in order to accurately determine the cost function gradients with respect to the ansatz parameters.

Since the standard python scipy.optimize.minimize implementation is used, the optimizers available here will only be used. This still gives a wide range of possibilities such as SLSQP, CG, L-BFGS-B for derivative based options and derivative free options such as: Nelder-Mead, Powell and COBYLA. After initial testing the SLSQP algorithm was selected as the primary optimizer for this work and unless stated otherwise, this algorithm is used. Other options such as SPSA and Adam optimizers, of the Qiskit class were also tested, but these were slower than a direct implementation of the SLSQP algorithm and were thus not used further.

3.8. Retrieving the Non-Scaled Solution

In most of the papers talking about VQLS only the solution proportional to the original solution of the classical problem is mentioned. The discretized Poisson equation is changed into the form of a linear system as $Ax = b$. To solve this on a quantum computer, where only normalized vector states are used this needs to be translated into $A|x\rangle = |b\rangle$. However, in the process of doing so $|x\rangle \neq x$, but $|x\rangle \propto x$ or $cA|x\rangle = b$. Logically from normalizing b, one obtains $b = |b||b\rangle$, where $|b|$ represents the normalization factor. The original linear system is rewritten as:

$$c \cdot A|x\rangle = |b| \cdot |b\rangle \tag{3.15}$$

By multiplying both sides with $\langle b|$ we get

$$c \cdot \langle b|A|x\rangle = |b| \cdot \langle b|b\rangle = |b| \tag{3.16}$$

due to the fact that $\langle b|b\rangle = 1$, since it is quantum state. By isolating c on the left hand side one obtains:

$$c = \frac{|b|}{\langle b|A|x\rangle} \tag{3.17}$$

By now substituting this c into $cA|x\rangle = Ax$ and dividing by A, the following result is obtained:

$$\frac{|b|}{\langle b|A|x\rangle}|x\rangle = x \tag{3.18}$$

Dividing by the A-matrix, is only possible if the problem is invertible, which is a general assumption of the VQLS algorithm. By defining $\lambda = \frac{1}{\langle b|A|x\rangle}$ the non-scaled solution is rewritten as $x = \lambda|b|x$.

The full vector x can only be retrieved when the full quantum state $|x\rangle$ is reconstructed by, for example, quantum state tomography. This in itself is an inefficient process and requires at least $\mathcal{O}(N)$ measurements. Doing so would only make sense to do once at the end of the algorithm and result in losing any advantage over classical methods. As mentioned in the original VQLS paper [10], the output can also be obtained in the form an expectation value $\langle x|M|x\rangle$. In this case it is also possible to use the non-scaled solution of the original problem in an efficient way by applying the scaling factor in front of this term. Meaning that an efficient output of the non-scaled solution can be used.

4 Methodology

In this chapter the different methods to implement the VQLS algorithm are discussed. These range from very simple linear algebra implementations to the complete quantum implementation as is shown in the previous chapter. First the problem at hand will be explained.

4.1. Discretization of the Poisson Equation

During this work the 1-dimensional Poisson equation with Dirichlet boundary conditions is solved. This equation is chosen as it is in principle a simple problem, however it becomes quite demanding to solve for increasing problem sizes. The Poisson equation is the most simple elliptic partial differential equation and appears in a wide variety of problems. The Poisson equation is defined as:

$$-\nabla^2\phi = f \tag{4.1}$$

where in the one-dimensional case $f = f(x)$. For the problem in this work, the function $f(x) = x$ is chosen on the domain [0,1]. This function is equal to the x coordinate and the domain [0,1] is chosen as it does not require any scaling. Dirichlet boundary conditions are used and are set to zero at both sides, such that the total problem is defined as shown in eq. (4.2)

$$-\nabla^2\phi = x, \quad \phi \in [0,1], \quad \phi(0) = \phi(1) = 0 \tag{4.2}$$

By using the central difference scheme, the second derivative of an arbitrary function $f(x)$ is approximated by eq. (4.3). Here h is the grid spacing defined as $h = 1/N$, where $N = 2^n$ is the number of nodes. By applying this to the Poisson equation a discretized form is obtained, which is then translated into a linear system of equations and is solvable by means of VQLS.

$$f''(x) \approx \frac{f(x-h) - 2f(x) + f(x+h)}{h^2} \tag{4.3}$$

By increasing the number of nodes and thus decreasing the value of h this approximation will become more accurate. Applied to the Laplace operator this results in eq. (4.4) in matrix form.

$$A = \frac{1}{h^2}\begin{bmatrix} 2 & -1 & 0 & \dots & & \\ -1 & 2 & -1 & 0 & \dots & \\ 0 & -1 & 2 & -1 & & \\ \vdots & & \ddots & \ddots & \ddots & \\ & & & -1 & 2 & -1 \\ 0 & 0 & 0 & \dots & -1 & 2 \end{bmatrix} \tag{4.4}$$

This matrix scales as 2^n in row and column size and $2^n x 2^n$ in the number of elements, where n is the number of qubits. Meaning that also for a classical solver, the required computational effort grows extremely quickly in the problem size as more qubits are used.

The right hand side vector of the problem is prepared by normalizing the vector b. Originally this is a continuous vector, but since in the linear system a limited number of nodes are used, it has to be discretized first. This is done by dividing the domain with equidistant spacing. Since the first and last nodes of the problem are fixed due to the boundary conditions, these are left out of the linear system. The total problem size is thus 2^n nodes, for n qubits. Once the discretized vector x is prepared, the normalized quantum state is formed by eq. (4.5):

$$|b\rangle = \frac{x}{|x|} = \frac{x}{\sqrt{\sum_i x_i^2}} \tag{4.5}$$

This assures that the x-coordinate vector is normalized and able to be transformed into a quantum state.

One thing to note in the definition of the A-matrix as in eq. (4.4), is the factor $\frac{1}{h^2}$ in front of it. Classically this factor is important, as it keeps a bound on the lowest eigenvalue. If the factor would not be used, the largest eigenvalue would be limited and the smallest would be ever decreasing for larger problem sizes. Since A is a symmetric matrix, one can use the spectral theorem to show that:

$$\lambda_{min} \leq u^T A u \leq \lambda_{max} \tag{4.6}$$

Here u is an arbitrary normalized vector. Since quantum states are also normalized vectors, this means that when evaluating an expectation value in the form of $\langle\psi|A|\psi\rangle$, where A is a symmetric (Hermitian) matrix, the smallest eigenvalue will indicate how small the output can get. Since on a quantum computer only normalized quantum states and Hermitian operators are used, the factor $\frac{1}{h^2}$ is of less importance. A decomposition in the form of $A = \sum_i c_i A_i$ is used. However, the accuracy of the result is determined by how accurate $\langle\psi|A_i|\psi\rangle$ is evaluated and any scaling of the c_i coefficient in front, scales both the expectation value and the error itself. Meaning that in the end only a normalized state $|x\rangle$ is obtained and the factor $1/h^2$ would only change the absolute value of the cost function and not affect the quantum state $|x\rangle$ itself.

4.2. Matrix Condition Number

The condition number is a widely used metric in scientific computing. It gives a measure for how much the output of a linear system changes, based on a small change in the input. The condition number can also be considered as a measure for how difficult a matrix is to invert, since a very large condition number means that a matrix is close to singular. The condition number is defined as shown in eq. (4.7). Here $\lambda_{max}(A)$ and $\lambda_{min}(A)$ are the largest and smallest eigenvalue of the A-matrix.

$$\kappa(A) = \frac{|\lambda_{max}(A)|}{|\lambda_{min}(A)|} \tag{4.7}$$

The condition number of the A-matrix gives a good indication of how difficult a general matrix is to solve, since it is a measure of how sensitive a linear system is to variations to the input. Since VQLS, as its name implies, solves a linear system, this is an important factor to consider. In fig. 4.1 the condition number of A is plotted versus the problem size in number of qubits.

There is a clear exponential relation between the problem size and κ. Even when the total vector size is considered (2^n states), the relation is still exponential. Meaning that even on a quantum system, the problem will quickly become very difficult to solve.

The last part of the previous section discussed the effect of the scaling in front of the A-matrix. When this scaling is included, the smallest eigenvalue reaches a certain minimal value, while the largest keeps increasing when the problem size is increased. When the scaling is left out of the picture, it is the other way around, where the largest eigenvalue reaches a limit and the smallest one keeps getting smaller. However, in both cases the condition number stays the same, irrespective of what scaling is used.

4.3. Adapted Poisson Equation

By looking at the progression of the condition number for larger problem sizes in the previous section, it is expected that the Poisson equation will quickly become difficult to solve as the problem size increases. In order to still be able to solve relatively large problem sizes, the Poisson equation can be

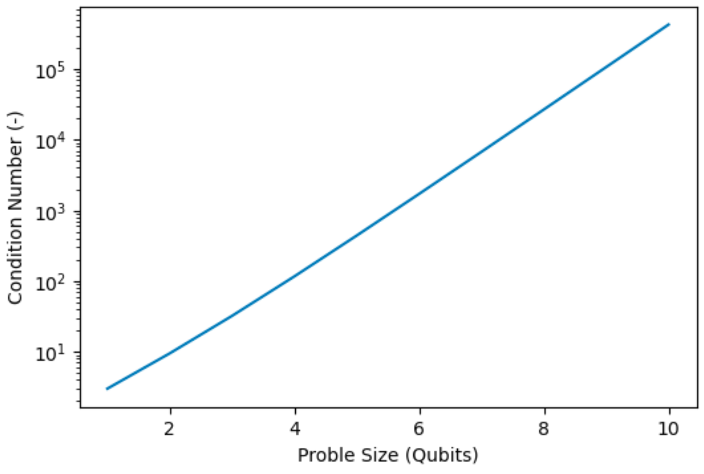


Figure 4.1: Condition number versus the problem size in number of qubits.

made easier to solve. This is done by adding an additional term on the left-hand side of the original problem as is shown in eq. (4.8). This has the effect of improving the conditioning of the A-matrix when discretized, however it also changes the problem itself, so a different solution will be obtained. The coefficient c can be any positive number, where a larger number has a better effect in reducing the condition number.

$$-\nabla^2\phi + c\phi = x \tag{4.8}$$

This extra term adds the identity matrix c times to the A-matrix of the original problem and this way changes the eigenvalues. Doing this puts a limit on the lowest eigenvalue equal to c. The largest eigenvalue will slowly reduce. This means that a limit forms on the maximum value the condition number can attain for this adapted equation. The condition number of the A-matrix of the adapted Poisson equation is shown in fig. 4.2. Here κ does not exponentially increase but rater approaches a limiting value for larger problem sizes.

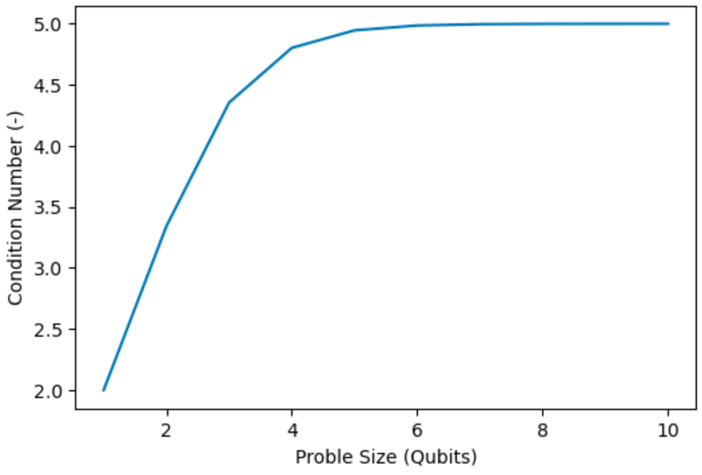


Figure 4.2: Condition number of the A matrix in $-\nabla^2\phi + c\phi$, where $c = 1$.

This would indicate that solving the adapted form of the Poisson equation could be easier than the original problem where $c = 0$.

Another similar idea which will be tested is morphing between the adapted form and the original equation. In the case where the adapted form with $c = 100$ or $c = 1$ would be easier to solve, it is possible to slowly reduce the value of c, such that finally the desired solution of the $c = 0$ case is obtained. This method does however introduce other problems such as how fast c should be reduced. This will most likely depend on the problem sizes and could be difficult to determine a priori. Another solution could be to add the coefficient c to the parameters in the optimizer and let the optimizer itself choose how to change it. Both cases will be tested and are covered in chapter 5.

4.4. Linear Algebra Implementation

With the aim of constructing an efficient fast working non-quantum implementation of the VQLS algorithm, a combination of a state vector simulator and basic linear algebra is used. The idea is to use Qiskit to prepare a quantum state based on the ansatz. As a next step this quantum state is converted into a classical vector and only linear algebra is used to evaluate the cost function. Since all information regarding A, A^2 and b is already known, this is very quick and easy to implement. This way there is also no need for the decomposition of the A-matrix as the resulting vector of a matrix-vector product does not longer have to be a quantum state. Thus $\langle\psi|A^2|\psi\rangle$ and $\langle b|A|\psi\rangle$ can be simply evaluated by doing matrix-vector products with the pre-computed tensors.

Since this method still relies on the use of the ansatz in a quantum circuit, it is possible to draw conclusions during the optimization process of the cost function. For example, if this method, with a number of significant simplifications, already struggles to solve the problem, then a full quantum implementation will most likely not perform better. Another benefit of having this simple algorithm is for verification purposes. By having this exact method, more complex methods can be compared to it later on. This can for example be used during the evaluation of all the individual decomposed terms.

Another advantage is that the implementation does not require the PSR, as all derivatives can be obtained by finite differences method, often implemented in the standard optimizers in Python.

4.5. State Vector Simulations

This form contains the quantum circuits as explained in chapter 3. More complexity is added over the previous method, since now the actual decomposition of A and A^2 is used. This means that more complex quantum circuits, with more parts than just the ansatz are used. The approach still relies on the StatePreparation method for generation of the b vector. In fig. 3.6 and fig. 3.7, the circuits are shown with this method. The purple unitaries with $|\psi\rangle$ and $|b\rangle$ are generate by the StatePrepartion method and not with actual unitaries in the form of $U(\theta)|0\rangle = |\psi\rangle$. This way also the implementation of the algorithm can be simply verified. Due to the fact that with this approach the probabilities are exactly known, the PSR is not needed. The standard optimizers used in Scipy optimize toolbox are both gradient-based as gradient-free and since the results from this implementation of algorithm are exact, no uncertainties are introduced during the evaluation of the gradients.

This approach is an intermediate step between the linear algebra approach on that of the quantum implementation and is a good test for the post-processing method as shown in section 3.5. This instance of the algorithm is verified by comparing the output of test runs to the exact results as obtained with the linear algebra method. If an exact replication of the results is obtained, it is concluded that this implementation also works correctly and the final step towards a full quantum implementation can be made.

4.6. Quantum Implementation

Finally, a full quantum implementation of the algorithm is used. In this case $|\psi\rangle$ and $|b\rangle$ are constructed with a parameterized ansatz, shot noise is used and the post-processing of the measurement data, as explained in section 3.5, is used. Which ansatz is used for the generation of $|\psi\rangle$ can be freely chosen, as it is a part of the algorithm that can be easily changed. The implementation of this algorithm is verified by using the exact method from the linear algebra implementation. Due to the fact that now shot noise is taken into account, the evaluation of the expectation values is no longer exact. This has a strong effect on how the gradients must be evaluated. Even for gradient-free optimizers, where the gradient is not explicitly used, one could expect that optimization would be less influenced by shot noise. However, these optimizers also have a very difficult time optimizing the cost function.

For the gradient-based optimizers a PSR function is constructed which evaluates the gradient of the cost function with respect to each parameter. This is done exactly as explained in section 2.11 and in section 3.6.1 where the quotient rule for evaluating the PSR with a normalized cost function is explained.

The addition of the PSR function makes the full quantum implementation computationally significantly more expensive. For each additional parameter in the ansatz the gradient must be evaluated by using at least two shifted angles, meaning that the number of circuit evaluations goes up drastically.

4.7. Adapted Cost Function

After initial testing, optimization of the standard and normalized cost function quickly becomes difficult as the problem size increases. In an attempt to improve the trainability of the ansatz, a different approach is taken for the cost function. Often in classical methods a quadratic loss function in the form of eq. (4.9) is used.

$$L = \arg\min_{x} |Ax - b|^2 \tag{4.9}$$

In order to implement this type of cost function on a quantum computer one has to take into account that the quantum state $|x\rangle$ is only proportional to the actual solution x. So a direct implementation of eq. (4.9) is not possible. In section 3.8, a method for re-scaling of the classical solution was introduced and this could potentially open up a method where the classical form can be used. The loss function can be converted to one suitable for a quantum computer as shown below.

$$(Ax - b)^2 = (Ax)^2 - 2bAx + b^2 \xrightarrow{Quantum} \langle x|A^\dagger A|x\rangle - 2\langle b|A|x\rangle + \langle b|b\rangle \tag{4.10}$$

$$L = \lambda^2 \langle x|A^\dagger A|x\rangle - 2\lambda\langle b|A|x\rangle + \langle b|b\rangle \tag{4.11}$$

Translating this into a cost function for the quantum implementation yields:

$$C = \lambda^2 \langle x|A^\dagger A|x\rangle - 2\lambda\langle b|A|x\rangle + 1 \tag{4.12}$$

where $|x\rangle = U(\theta_{opt})|0\rangle$, λ is the scaling coefficient used in retrieving the original non-scaled solution and is equal to $\lambda = \frac{1}{\langle b|A|x\rangle}$. Using this additional term λ as an extra parameter in the optimizer might be able to outperform the standard or normalized cost functions.

An interesting point to note is that if $\lambda = \frac{\langle b|A|\psi\rangle}{\langle\psi|A^\dagger A|\psi\rangle}$ is used, the cost function as in eq. (4.12) translates to the normalized cost function.

Another cost function which is tested, is one where the fidelity is used as a measure of overlap between the quantum state prepared by the ansatz and the solution vector $|x\rangle$. Mathematically this results in a cost function of the form $C(\theta) = 1 - |\langle x|\psi\rangle|^2$, where $|\psi\rangle = U(\theta)|0\rangle$. The fidelity is subtracted from 1, to make sure that the solution is found when $C = 0$. In practice, using the fidelity as a cost function is unfeasible to implement on a quantum computer due to two different reasons. The first is that it already requires the solution in order to match the ansatz to this exact state. The second is that one also already must know how to implement it with an ansatz. Knowing the exact solution vector $|x\rangle$ is one challenge, but knowing how to construct it on a quantum computer makes it even more challenging and this is exactly what is already required for the evaluation of $\langle x|\psi\rangle$. However, using such an approach to see if the ansatz is trainable does give useful insights in the capabilities of the classical optimizer in a somewhat ideal scenario. It shows whether an ansatz is capable of fitting to the given solution and is comparable to a type of supervised learning. If this is not possible, it can be concluded that standard cost function will also be difficult to use.

4.8. Extracting Quantum Information

Another difficulty of the VQLS algorithm is efficiently extracting useful information out of the quantum state once the cost function has been optimized. In the original VQLS paper [10], the proposed approach to efficiently extract quantum information out of the system is by evaluating an expectation value in the form of $\langle x|M|x\rangle$, where M is a measurable operator. However, finding a suitable operator M which actually gives useful information is not trivial.

It is possible to efficiently extract the amplitude of a single position in the domain. By using the Swap-test it is possible to efficiently evaluate the fidelity in the form of $\langle v_{ref}|\psi\rangle$. By generating a reference state, which has zero amplitude everywhere and an amplitude of 1 corresponding to the position where the solution is desired to be evaluated, it is possible to extract this information. An example of such a vector is $v_{ref} = (0, 0, 1, 0)$ for the case of 2-qubits. By then evaluating the fidelity $\langle v_{ref}|\psi\rangle$ a measure of overlap between the quantum state, prepared by the optimized ansatz and this newly created reference state, the amplitude at this position is obtained. This can be done with the Swap-test as explained in section 2.3. The circuit for a 2-qubit case is shown in fig. 4.3. Here the $|\psi\rangle$-state, on qubits 1 and 2, is the solution vector $|x\rangle$ obtained after optimizing the ansatz and the second state $|v_{ref}\rangle = (0, 0, 1, 0)$ on qubits 3 and 4 is the reference state. By running this circuit a certain number of shots, an estimate for $|\langle v_{ref}|\psi\rangle|^2$ is made. Since it is known that the amplitude will be a positive value, one can take the square root of this number to obtain $\langle v_{ref}|\psi\rangle$.

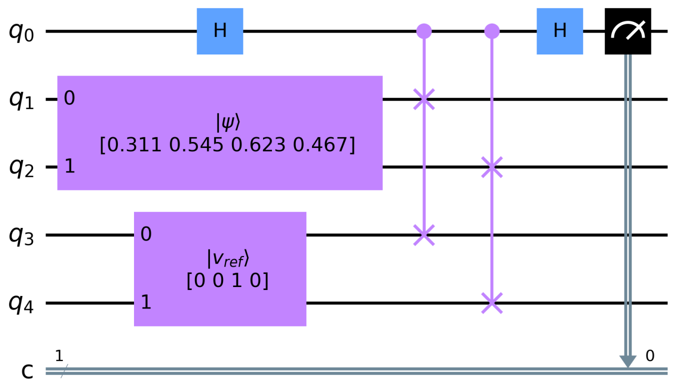


Figure 4.3: Example of the Swap-test of for the 2-qubit case. The purple boxes for $|\psi\rangle$ and $|v_{ref}\rangle$ contain the unitaries to generate the quantum states shown in the respective box.

Only obtaining a single amplitude this way might not seem that useful, however one can use other properties of the solution to gain more information out of it. For example, the solution to $-\nabla^2\phi = x$ has its maximum around the $x = 0.6$ location along the domain. By using the fidelity with the quantum state closest corresponding to this position, it is possible to extract the maximum amplitude at this position. Since the value obtained with the above method is a normalized one, it can be re-scaled to the classical maximum value by using $\phi_{max} \cdot |b| \cdot \lambda$. In the case of the Poisson equation solved here, a third order polynomial can construct almost an exact fit over the classical solution, by using the 4 known data points: $\phi(0) = 0$, $\phi(1) = 0$, $\phi(x_{\phi_{max}}) = max(\phi)$ and $d\phi(x_{\phi_{max}})/dx = 0$. $max(\phi)$ is obtained with the Swap-test as explained above. With this approach, it is theoretically possible to reconstruct something very similar to the classical solution, by only using one single amplitude of the quantum state.

This approach is of course problem dependent and will not work for any arbitrary problem. However, for this instance of the Poisson equation it can serve as a solution for efficient extraction of data out of the quantum state in order to generate useful data classically.

Another reference vector which can be used to extract useful information, is the right hand side vector $|b\rangle$. This way one obtains a measure of energy when computing the overlap $\langle b|x\rangle$. As the number of nodes in a problem increases, this measure of energy should converge to a fixed value. This measure is not necessarily problem specific and might allow for a more general applicable for of date extraction.

4.9. Workflow of Solving the Poisson Equation

Now that the approach is layed out, a simple workflow of the VQLS algorithm is given.

1. Determine the type of Poisson equation that is being solved, the boundary conditions used, the right hand side vector and the problem size (number of qubits). Once this is obtained the system must be discretized to go from $-\nabla^2\phi = f$ to $Ax = b$.

2. Decompose the discretized A matrix (and A^2 matrix if required) in a linear combination of normal operators. For each decomposition a list of terms and corresponding coefficients must be

obtained.

3. Normalize the right hand side vector and be able to reconstruct it as a quantum state $U(\theta)|0\rangle = |b\rangle$.
4. Select an ansatz to prepare $U(\theta)|0\rangle = |\psi\rangle$.
5. Select the number of shots required to evaluate the expectation values, the cost function and the desired optimizer. If shot noise is used in combination with a gradient-based optimizer, the PSR rule is required.

6. Generate angles for the ansatz used as the initial guess for the optimizer. The next sub points show the internal loop in the optimizer:
   - 6.1 For each term in the decomposition of A and A^2 prepare the quantum circuits to evaluate $\langle\psi|A_i^2|\psi\rangle$ and $\langle b|A_i|\psi\rangle$.
   - 6.2 Obtain measurement data for the given number of shots.
   - 6.3 Apply post processing of the data in order to obtain the expectation value for each individual term.
   - 6.4 Add all terms together to obtain $\langle\psi|A^2|\psi\rangle = \sum_i\langle\psi|A_i^2|\psi\rangle$ and $\langle b|A|\psi\rangle = \sum_i\langle b|A_i|\psi\rangle$.
   - 6.5 Compute cost value.
   - 6.6 Continue onto next iteration of the optimizer until $C = 0$ or the maximum iteration number is reached.
7. When the optimizer is finished, check if it converged sufficiently. If it converged properly continue to the next step. If not, change hyperparameters such as:
   - Number of shots used for the evaluation of the expectation values
   - Cost function
   - Optimizer
   - Ansatz
   - Method of Decomposition
8. **Optional:** Extract solution data and use $U(\theta_{opt})|0\rangle = |x\rangle$ to generate the solution state from the optimizer parameters.
   - Perform quantum state tomography to reconstruct quantum state $|x\rangle$ classically.
   - Use an operator M to evaluate an expectation value in the form of $\langle x|M|x\rangle$.
   - Use the Swap-test to extract energy or amplitude data in the form of $\langle v_{ref}|x\rangle$.

5 Results

This chapter covers the presentation and discussion of the results obtained with the previously explained methods. First the classical solution is introduced, after which the results obtained with the linear algebra implementation are shown. Due to the fact that this implementation took far less time to run than the simulated quantum implementation, most results shown are obtained with this method. Then the cost function landscapes and the variance of the gradients are layed out. This then leads to a discussion on what influences the shape of these landscapes. Finally, the results of different cost functions and run times of the different implementations are shown.

5.1. Classical Solution

To start off, first the classical (analytical) solution to the Poisson equation is given. The analytical solution to $-\nabla^2\phi = x$, is obtained by simply integrating and using the boundary conditions $\phi(0) = 0$ and $\phi(1) = 0$. This results in $\phi(x) = -\frac{1}{6}x^3 + \frac{1}{6}x$. It is also possible to directly find the solution vector by constructing the A-matrix, the b vector, setting up the linear system $Ax = b$ and inverting the A-matrix in a software such as Python. In fig. 5.1 the solution vector is shown for 2, 4 and 8-qubit problems. Here the solution vector is normalized such that it will correspond to what is found as the normalized quantum state $|x\rangle$ when using VQLS. As a result of the solution being normalized, the amplitudes will decrease for increasing problem size. This already hints at a problem for the quantum implementation of the VQLS algorithm. In order to correctly evaluate a quantum state, an increased order of accuracy is required. In a later section of the results this is discussed in more detail.

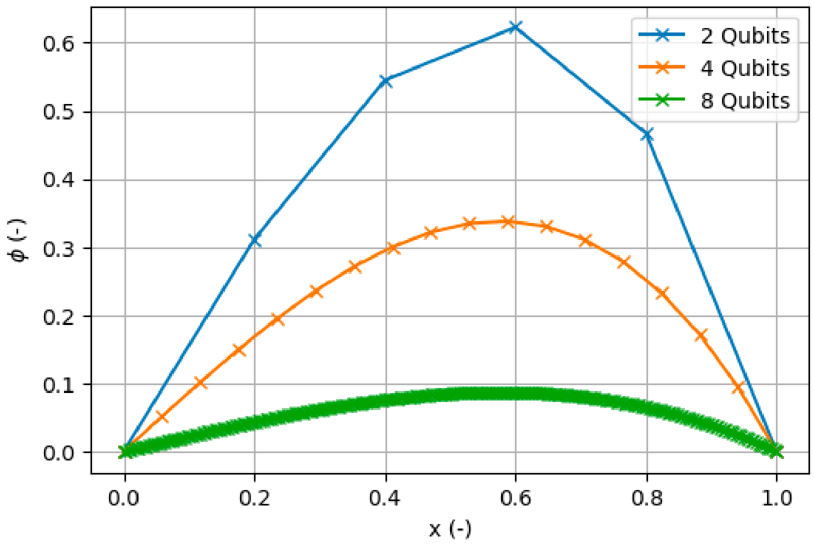


Figure 5.1: Normalized analytical solution for 2, 4 and 8 qubit problems (4, 16 and 256 nodes).

Due to the fact that the right hand side vector $f(x) = x$ is asymmetric, the final solution is also asymmetric. In the case where a symmetric b such as $b_i = 1/N^2$ would be used, the solution becomes symmetric, which can be easier to solve for certain ansatze. Now that the classical solution is established, it is time to see whether obtaining these results by means of VQLS is also possible.

5.2. Linear Algebra Approach

The first implementation of the VQLS algorithm is done by first evaluating the exact quantum state prepared by the ansatz, and then computing the cost by applying linear algebra as discussed in section 4.4. This simple and quick method allows for easy testing of new concepts, without having to wait a long time before results are ready. Even though this approach might not seem to add much compared to the complete quantum implementation, it will indicate how difficult it is to train the ansatz, as all other types of difficulties, such as shot noise and decomposition of the matrices, are removed from the problem with this simplified method. Once the VQLS algorithm was implemented, it was first verified by comparing it to results from literature.

5.2.1. Reconstruction Results Liu et al.

In order to test the linear algebra implementation of the VQLS algorithm, an attempt is made to reconstruct the results of Liu et al. [44]. The results from the linear algebra method using the same QAOA as used in their paper, are shown on the left in fig. 5.2, while the results by Liu et al. are shown on the right in fig. 5.3. Here on the y-axis, the fidelity with the true solution vector $|x\rangle$ is shown. For a small number of qubits, the curves look very similar. However for the 5 and 6 qubit case, the results by Liu et al. have a higher fidelity compared to what is shown on the left. This can be a result of various factors. Their algorithm does not take shot noise into account, which is a big factor when evaluating a quantum circuit. However, this is also not implemented in the method used to obtain the results in fig. 5.2. A reason for the differences could be the fact that they relied on a different classical optimizer. Initially the same BFGS optimizer was used, however the success rate of obtaining high fidelity solutions with this optimizer was relatively low and the SLSQP optimizer proved to work better. This indicates that there are some differences between the two implementations, as for them apparently the BFGS optimizer did work. Furthermore, their algorithm is written in ProjectQ, which is another open-source software framework in Python for quantum computing. The coding of the current work is performed using Qiskit, which is vastly different. It should not be expected that this is the cause for any differences, as the implemented code should work identically. However, it could be possible that due to slight differences in the implementation of state vector simulations between ProjectQ and Qiskit, discrepancies occur. Finally, optimization for the implementation of the current work had trouble optimizing problems larger than 4 qubits. With different hyperparameters, it is expected to have a better matching figure.

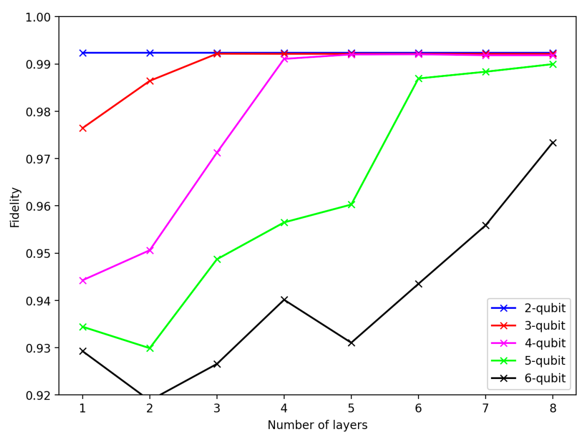


Figure 5.2: Results with linear algebra implementation.

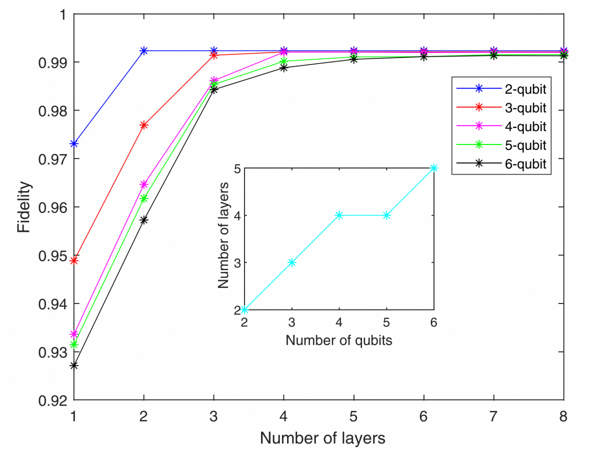


Figure 5.3: Results by Liu et al. [44]

In the figure it can be seen that the fidelity reaches a maximum of around $F = 0.992$. This is due to the fact that the QAOA, which they introduced in their work, is only capable of generating symmetric quantum states. The maximum of $\langle x|\psi\rangle$, where $|x\rangle$ is the normalized solution vector and

$|\psi\rangle = U(\theta_{opt})_{QAOA}|0\rangle$ is the quantum state generated by the ansatz, turns out to correspond to this value of $F = 0.992$. So Liu et al. use an asymmetrical ansatz to solve a symmetrical problem, which can be a questionable practice.

5.2.2. HEA Solutions

Now that it is confirmed that the implementation of the algorithm is working, the next step is to verify which problem sizes can be accurately solved. Rather than QAOA now the HEA, as is shown in section 3.1, is used. Again the linear algebra method is used to obtain these results, so no decomposition or any form of noise is taken into account. First, for each problem size the normalized classical solution $|x\rangle$ is computed. Which is used to evaluate the fidelity $\langle x|\psi\rangle$ between $|x\rangle$ and the quantum state obtained with the optimized ansatz $U(\theta_{opt})|0\rangle = |\psi\rangle$. This is done to get an indication of how many layers are required in the ansatz for each problem size and how difficult the ansatz is to train. In order to find the required number of layers, for each problem size first only a single layer is used. The next step is to train the ansatz for 50 different random initialization of the ansatz parameters and keep track of the highest fidelity achieved. If the achieved fidelity is not close to $\langle x|\psi\rangle = 1$, another layer is added and the process is repeated. By doing this a figure in the form of fig. 5.4 is obtained. The results are shown for 2 to 6 qubit problems, the normalized cost function is used in combination with the SLSQP optimizer. Here the PSR is not used, as all evaluations are exact and the standard built in methods for the gradient evaluations are used. 50 random initialization with angles between $-\pi$ and $+\pi$ are used for the ansatz and the highest fidelity after optimization is shown in the figure. Using the standard cost function for these tests did not make any significant difference, as the expressibility is only depending on the ansatz itself. The figure clearly shows that, as the problem size is increased the number of layers required for achieving a certain fidelity also increases.

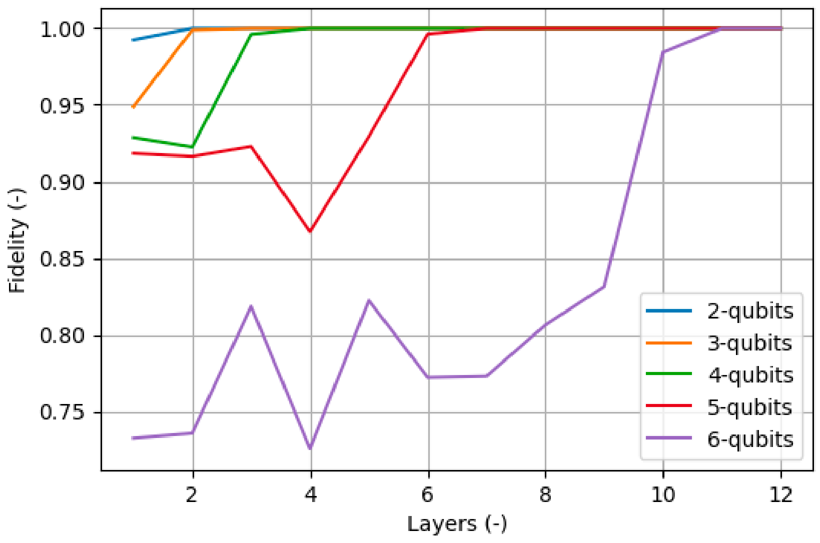


Figure 5.4: Highest achieved fidelity $\langle x|\psi\rangle$ between the normalized classical solution and the quantum solution versus the number of layers for the HEA.

In the figure it is also seen that for the larger problems, the number of layers and the maximum fidelity initially does not necessarily increase. For example, in the case of 6 qubits and 4 layers the highest fidelity obtained after optimization of 50 random starting points, was below $F = 0.75$. While for the same problem with only 3 layers it was around $F = 0.82$. In theory this should not be the case, as adding more layers of this specific ansatz increases the expressibility. This can indicate that the optimizer has trouble finding good results when less than sufficient layers are used and that adding more parameters makes the training of the ansatz parameters more difficult.

Another similar finding, which does not become clear from fig. 5.4, is that when an ansatz is used with less than sufficient layers required to obtain the true solution, it is very difficult to find the optimal solution. For example, with 5 qubits and 6 layers it is possible to achieve a fidelity of $F = 0.999$. However, with a random initialization, the optimizer had a success rate of achieving this less than 1%

of the time. While for the same 5 qubit problem with 8 layers the success rate is around 85% out of all random initializations. For this case more than 50 runs where performed in order to get more accurate percentages.

5.2.3. Number of Layers Versus Problem Size

Now that the number of layers required to reach a certain fidelity is obtained, the next logical step is to look how this scales with problem size. Ideally the number of parameters in the ansatz should scale logarithmically, while the number of nodes scales exponentially. Literature has shown that only ansatze which scale logarithmically, in combination with a local cost function, are guaranteed to avoid barren plateaus. In fig. 5.5, the number of layers and parameters required to reach $\langle x|\psi\rangle = 1$ is plotted versus the number of qubits. As a point of reference the number of nodes in a problem for each respective qubit size is also shown. Here it becomes clear that the required number of parameters almost scales identically to the number of nodes in a problem. Since only problems between 2 and 6 qubits are shown, it is of course uncertain of how this would scale for larger problem sizes, but it is most likely that trend would continue. If this actually is the case then this specific instance of the HEA would scale exponentially in the number of parameters and thus not be suitable for the avoidance of barren plateaus.

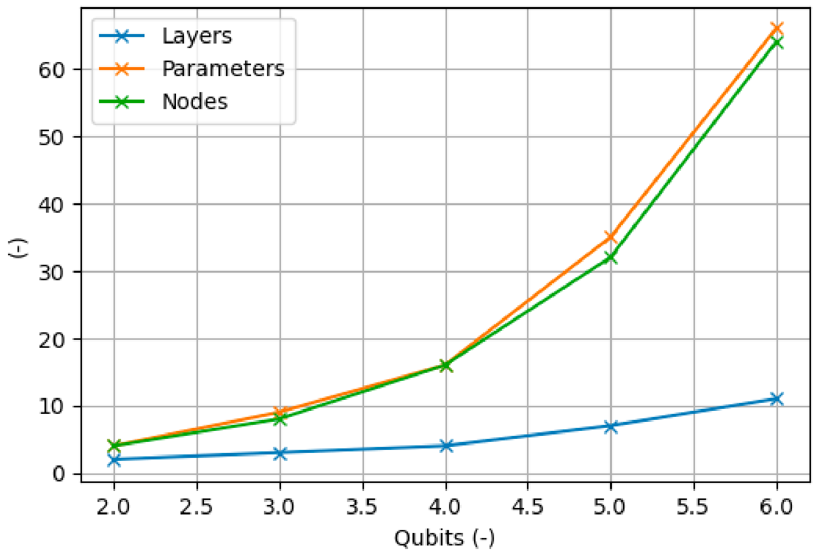


Figure 5.5: Number of required layers, parameters and nodes versus the number of qubits. The number of layers and parameters are the minimum to achieve an exact replication of the solution state $|x\rangle$.

Of course, for different ansatze this scaling can look completely different. Since the version of the HEA used here is one of the most general ansatze, it is useful to see how it scales. An ansatz specifically designed for solving this instance of the Poisson equation is likely to achieve a more favourable scaling in problem size. However, designing such an ansatz is not a trivial task and would be problem specific. Meaning that a slightly different problem could be incompatible to solve with that ansatz.

5.3. Results Quantum Implementation

From the results shown above, it was already assumed that the trainability of the ansatz with the quantum implementation, will most likely be even more of a challenge. After initial testing, it can indeed be confirmed that this is the case. A 2-qubit problem is doable to solve on a quantum device. However, the trainability of the ansatz quickly starts to become a challenge as the number of qubits is increased. For a 2-qubit problem, often times it is possible to achieve a fidelity in the order of $F = 0.999$ for both the standard and normalized cost function, when 10,000 shots are used per evaluation for the expectation value of the decomposed terms. An example is shown in fig. 5.6 for the 2-qubit case and the HEA with 2 layers, resulting in a total of 4 parameters. Here 10,000 shots are used during the optimization process and a 100,000 shots are used to reconstruct the final quantum state. This result is achieved with the normalized cost function and the SLSQP optimizer using the PSR rule. This results could as well have been shown for the standard cost function and any other type of optimizer, as the 2-qubit case is solvable for a wide variety of hyperparameters. For example, similar results have been achieved with the gradient-free COBYLA optimizer, which does not directly evaluate the gradients and thus does not rely on the use of the PSR. Of course also a larger number of shots could have been used to achieve a higher fidelity, however this comes at the cost of longer run times.

On the right in fig. 5.7 a result for a 3-qubit problem is shown. These results are obtained with the HEA with 3 layers and 9 parameters in total. During the optimization procedure 100,000 shots are used in combination with the normalized cost function. The final state as shown in the figure is reconstructed with 100,000 shots. Again the PSR is used in combination with the SLSQP optimizer. This time a very high fidelity with the normalized classical solution of $F = 0.999$ is achieved. This can partly be attributed to the increased number of shots, but also due to luck with the random initialization of the ansatz parameters. A large amount of these 3-qubit optimization procedures where performed and most of them ended at a fidelity of around 0.8-0.95, meaning that this result was exceptionally high. Nonetheless, it shows that it is indeed possible to solve the 3-qubit problem on a quantum computer. The results for the 3-qubit case are obtained with the gradient based SLSQP optimizer. Other tests were done with the gradient-free COBYLA optimizer, however this optimizer was not able to significantly reduce the cost function and achieve any form of meaningful optimization.

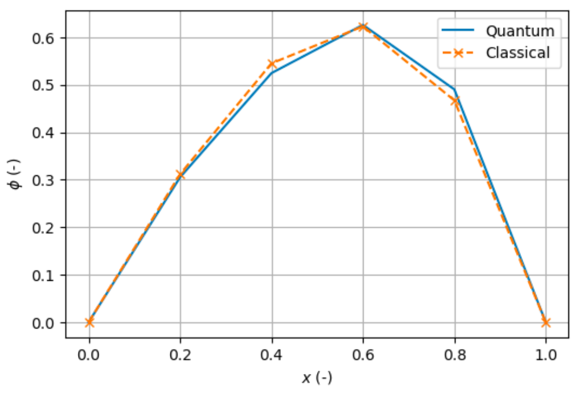


Figure 5.6: Result of a 2-qubit problem with the quantum implementation. During optimization 10^4 shots are used, while the final quantum state is reconstructed with 10^5 shots. Fidelity is $F = 0.999$.

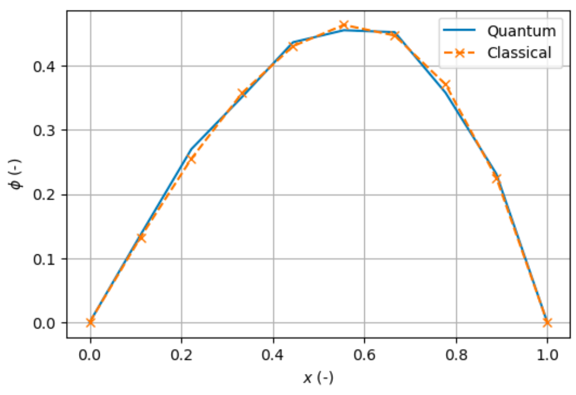


Figure 5.7: Result of a 3-qubit problem with the quantum implementation. During optimization 10^5 shots are used. The final quantum state is also reconstructed with 10^5 shots. Fidelity is $F = 0.999$.

Attempts for larger qubit problems did not achieve such good results as shown above. The 4-qubit problem is expected to require a number of shots in the order of 10^6-10^7 per term evaluation, which becomes very expensive computationally. Several attempts for optimizing this problem with 4 layers of the HEA and 1,000,000 shots have been done and the best result is shown in fig. 5.8. This is obtained with the standard cost function and the gradient based SLSQP optimizer. The final value for the cost function was $C = 0.02$, which corresponds to a fidelity of $F = 0.883$. Considering that a random initialization of the standard cost function often results in a value between $C = 1 - 10$, a considerable amount of training has happened to the ansatz. The quantum state obtained after optimization, does not exactly correspond to the normalized classical solution $|x\rangle$. If one would only consider the value of the cost function $C_{standard} = 0.02$, it might seem as if the optimization has succeeded, but this does not give any guarantee on whether a high fidelity with the true solution is achieved. In this instance of the optimization the values of the two terms in the cost function are $\langle\psi|A^2|\psi\rangle = 0.02421$ and $|\langle b|A|\psi\rangle|^2 = 0.00415$. In the normalized cost function this corresponds to $C_{norm} = 0.829$, which gives a better indication at the true convergence, considering it on a scale between 0 and 1. A single optimization costs several hours for these problem sizes and the majority of the time waiting is spent at evaluating all the terms while computing the gradients with the PSR. From these results it was concluded that solving even larger problems would not be successful and thus not attempted.

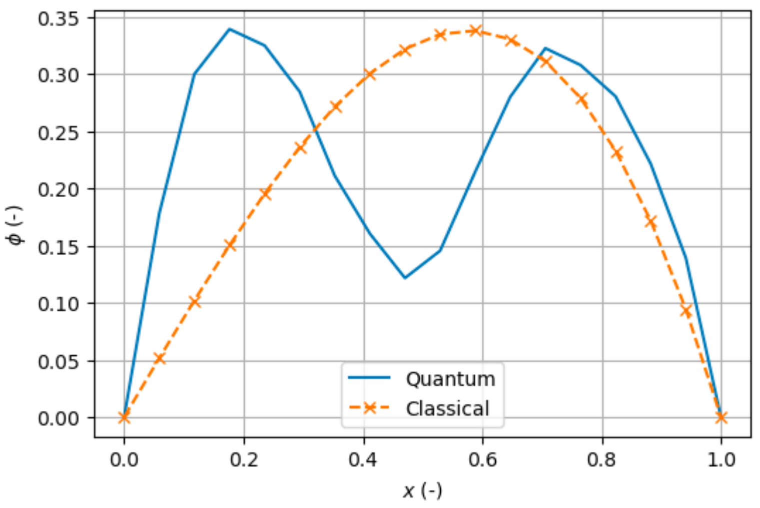


Figure 5.8: Result of a 4-qubit problem with the quantum implementation. During optimization 10^6 shots are used. The final quantum state is also reconstructed with 10^5 shots. The final value of the standard cost function was $C_{standard} = 0.02$, while the fidelity is $F = 0.883$,

An interesting point to note about the quantum implementation, is that when exact probabilities instead of shots are used (i.e. an infinite number of shots), results are exactly the same as with the linear algebra implementation. For example, when the simplified method and the quantum implementation with exact probabilities were both initialized with the same angles and optimizer, the final results would be identical. This also indicates that the quantum implementation is correct and that the simplified model can be used to obtain results faster and try out different approaches in order to save time.

5.4. Cost Function

The main difficulty while solving the Poisson equation by means of VQLS is the trainability of the ansatz. As shown in the previous section, optimization of the 2 and 3-qubit problem is doable. While for 4 or more qubits, it practically becomes impossible unless a very large number of shots is used. The trainability of the ansatz primarily depends on the so-called cost function landscape. If this landscape has a smooth shape with large gradients, the cost function is easy to optimize, but if the gradients are small it can become a very difficult task. A logical next step is to investigate what this cost function exactly looks like and what determines the shape of its landscape.

5.4.1. Cost Function Landscape

To get a better understanding of why optimization of the cost function is so difficult a reconstruction of the cost function landscape is made. The cost function landscape is a k-dimensional landscape, where k is the number of parameters in the ansatz. By only selecting two out of all the parameters, this landscape can be visualised in three dimensions by having the cost on the third axis. Due to the cyclic nature of the rotation gates in the ansatz, when plotting these two angles in the range between $-\pi$ and $+\pi$ on the x and y-axis and plotting the cost on the z-axis, a periodic shape of the cost function landscape is obtained. First the optimized ansatz parameters are found which correspond to a quantum state where $C(\theta_{opt}) = 0$. Then only the first two parameters are varied, while the rest is kept constant. For both the standard and normalized cost functions, different combinations of the two parameters which are plotted on the x and y-axis were tested, but this did not make any significant differences to only using the first two ansatz parameters. The general shape of the landscape is very similar for any two combinations of ansatz parameters.

What this cost function landscape looks like for 2, 3 and 4-qubits is shown in fig. 5.9 for the standard cost function and in fig. 5.10 for the normalized cost function. For these results again the HEA ansatz is used with 2, 3 and 4 layers for the respective 2, 3 and 4-qubit problem sizes. Furthermore, these results are obtained with the linear algebra approach. Similar plots have also been constructed with the quantum implementation and showed identical landscapes. The figures are made by discretizing

θ_i and θ_j in 40 different nodes, resulting in a total of 1600 points to construct the landscape. For the standard cost function it is clear that there are large gradients present. For all cases the landscape remains 'hilly' with large and smooth gradients. However, looking from case to case it does become clear that the maximum of the landscape reduces. It might seem as if the cost function is easy to minimize with such a landscape, however it is still a very difficult task. This is due to two reasons. First, the landscape suffers from local minima, making it very difficult to find the true global minimum. Secondly, the landscape close to the solution has a very flat shape, meaning that even when one is close to the actual solution, it is still very difficult to exactly find the angles corresponding to the true solution.

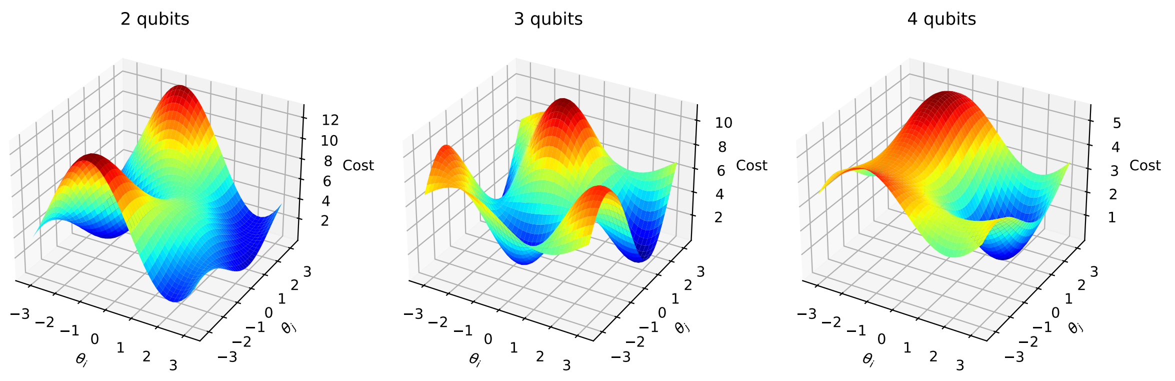


Figure 5.9: Landscape of the standard cost function for 3 different problem sizes.

The landscape for the normalized cost function as shown in fig. 5.10, looks completely different from the standard cost function. Rather than having large gradients, here the majority of the landscape is completely flat and the solution lies at the bottom of a deep well. This immediately reminds of the phenomena of barren plateaus. Barren plateaus are defined as regions in the cost function landscape where the variance of the gradients vanishes exponentially in the number of qubits, which seems to be the case here. For the 2-qubit problem, the gradients further away from the solution are still clearly pointing towards the well. However, for the 3-qubit problem this is already much less and for the 4-qubit problem most of the landscape looks completely flat, apart from very close to the well. This phenomena only becomes worse for larger problem sizes.

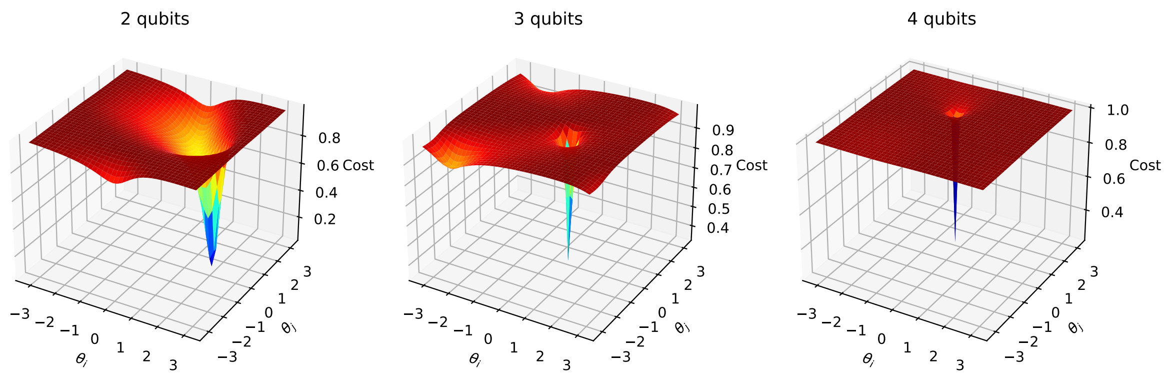


Figure 5.10: Landscape of the normalized cost function for 3 different problem sizes.

The results shown above were all obtained with exact simulations, however when one was to implement VQLS on a NISQ device, noise is introduced. As expected, noise makes the whole optimization process much more difficult. Figure 5.11 shows a noisy cost function landscape for a 4-qubit case. Here only 100 shots are used to exaggerate the effect of shot noise. Normally, one would use a much larger number of shots, but in that case the effect would not be as clear on a figure like this. Nonetheless it does show why optimizing with noise is more complex. In exact conditions the landscape away from the gorge is **almost** flat, but there are still small gradients present which can be used to find the true minimum. However, when noise enters the picture, the amplitude of the noise can be significantly larger than the small gradients that are present, meaning that optimization effectively becomes impossible. The only way to circumvent this is by increasing the shot number such that the shot noise is smaller than the value of the gradient.

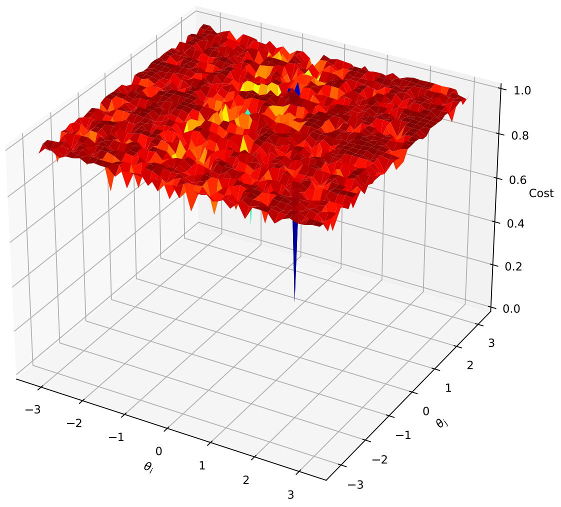


Figure 5.11: Noisy landscape of a 4-qubit problem. Here only 100 shots are use in order to better show the effect of noise on the landscape.

The above discussion does only mention the effect of shot noise. During hardware runs on a quantum device, various other forms of noise are introduced such as implementation error when rotating qubits or quantum decoherence. These types of noise will not be reduced when increasing the number of shots and solutions must be found on the hardware side of quantum computing. During optimization, this has the effect that the landscape becomes even more noisy and distorted, meaning that optimization becomes even more of a challenge.

5.5. Vanishing of the Gradients

In order to investigate the phenomenon of vanishing gradients and barren plateaus, the variance of the gradient versus the number of qubits and number of layers is investigated. The variance of the gradients in the cost function is used as a measure of barren plateaus and gives a good indication of how 'flat' a higher dimensional cost function landscape is. In order to construct such a figure, 200 random samples are taken and the gradients of the cost function are evaluated with respect to a certain ansatz parameter. The variance over these 200 gradients is computed and plotted versus the number of qubits. The results are shown in fig. 5.12 and are obtained with the linear algebra method, where the gradients are evaluated by using the PSR. This means that an exact value of the gradients is obtained and no shot noise is present. In the figure on the left, results are shown for the standard cost function and on the right for the normalized cost function. For all results the HEA, consisting of two layers, is used. As can be seen in the figure for the standard cost function, it has a relatively large variance of the gradient and a second order fit is very suitable to approximate the reduction in the variance. This means that the vanishing of the gradient in problem size is not exponential, as what is used in the definition of barren plateaus. Even for a 7-qubit problem the variance is still valued in single digits.

For the normalized cost function this is again a completely different story. It should be noted that the y-axis is plotted on a log scale. Here a first order curve fit is applied, which slopes down with a value of -2.82. Meaning that for each additional qubit, the variance of the gradients is reduced by around a factor of $e^{2.8} \approx 16$. Knowing this, it is logical why the trainability of the ansatz reduces so quickly in problem size. For the 4-qubit problem the variance in the gradient is already more than to two orders of magnitude smaller than for the 2-qubit problem. For both cost functions the results have also been obtained while changing the ansatz parameter at which the gradient is evaluated. This does not have a noticeable effect on what the general trends look like.

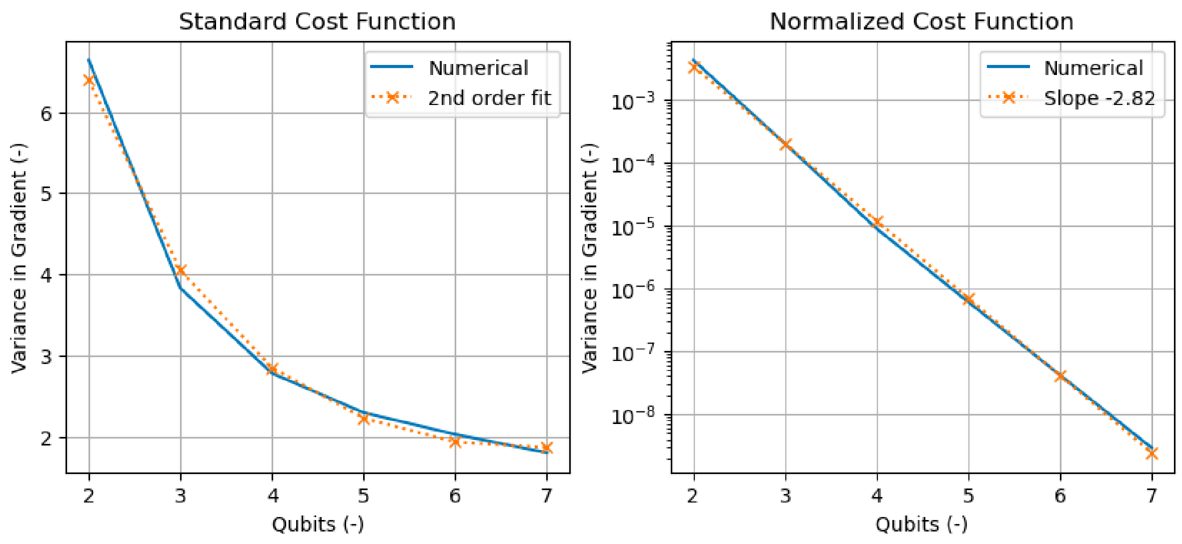


Figure 5.12: Variance of the gradient versus the number of qubits for the two cost functions. Two layers are used for all problem sizes.

The results above only show the changes of variance of the gradients in the number of qubits. But another interesting piece of information to look at, is seeing how the variance scales in the number of ansatz layers. For each increase in problem size, additional layers in the ansatz are required in order to achieve sufficient expressibility of the ansatz, in order for it to be able to construct the desired quantum state. Any increase in the ansatz expressibility is expected to be detrimental to the variance of the gradients, as it requires the complexity and depth of the quantum circuit to be larger. In fig. 5.13 the results are shown for the 2-qubit case and in fig. 5.14 for the 6-qubit case. The 2-qubit case shows that adding more layers does not make changes to the variance of the gradients for both the standard and normalized cost function. Changing the parameter at which the gradients are evaluated does not make a noticeable difference in the shape of the curves.

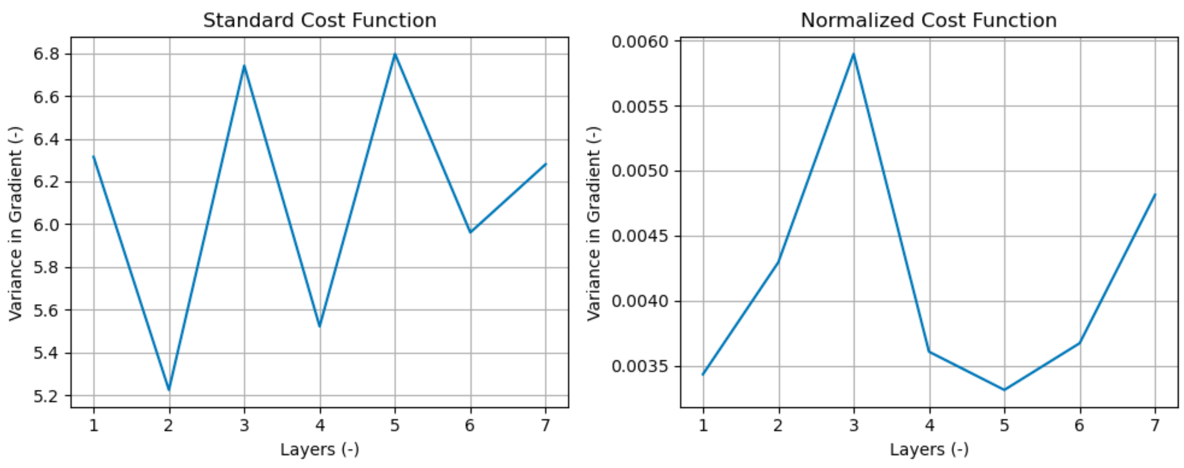


Figure 5.13: Variance versus the number of layers for the standard and normalized cost function. 2 Qubits are used for all layers.

For the 6-qubit case in fig. 5.14, it is clearly shown that there is a downward sloping trend as the number of layers increases for both the standard and the normalized cost function. A second order fit is added to both figures, to indicate at which rate it is sloping downward. From these two figures it seems as if for the 6 qubits case, the variance of the cost function gradient would continue to slope downward as more and more layers are added. However, it does seem to asymptotically reach a limit.

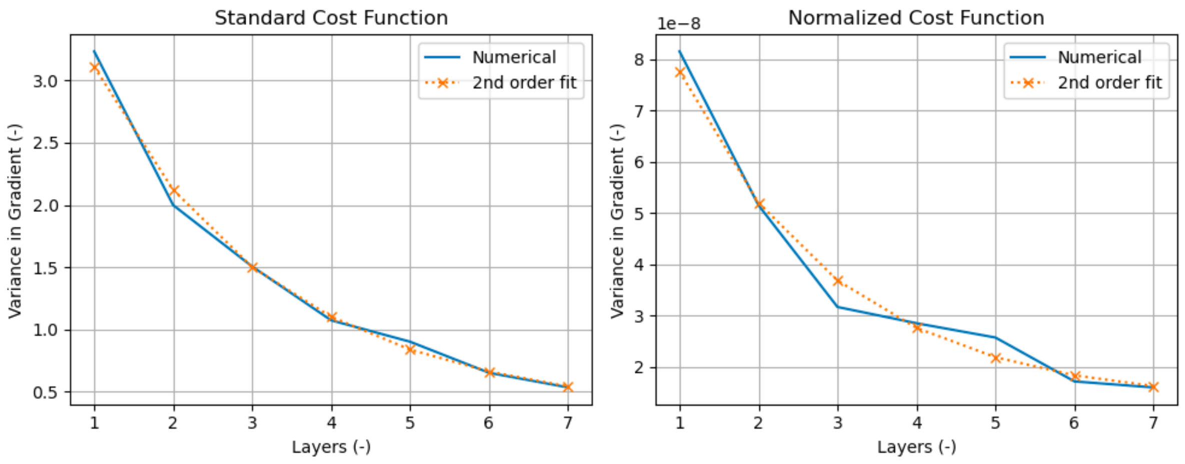


Figure 5.14: Variance versus the number of layers for the standard and normalized cost function. 6 Qubits are used for all layers.

The results obtained in the figures above do indicate that the reduction of the variance in gradients is only present for larger problem sizes. However, looking at the required number of layers for the ansatz to reach a sufficient level of expressibility for the 2 and 6-qubit cases, there might be a different reason behind it. For the 2-qubit case only 2 layers are required in the ansatz in order to construct the normalized classical solution. For the 6-qubit case, this value is 11 layers. It might be the case that after the required number of layers is reached, the variance of the gradients does not significantly reduce when more layers are added. For the 5-qubit case, 6-7 layers are needed. In fig. 5.15 the same figure is shown, but now for a 5-qubit problem. Here the trend is more clear for the standard cost function, but is also present in a milder form for the normalized cost function. It can be seen that initially the variance starts out at a relatively large value and as the number of layers is increased, it reaches a certain lower limit. This seems to be achieved around 7-8 layers, which is also around the limit where the ansatz becomes the best trainable. When more layers are added after this point, the variance of the cost function gradient seems to not decrease further.

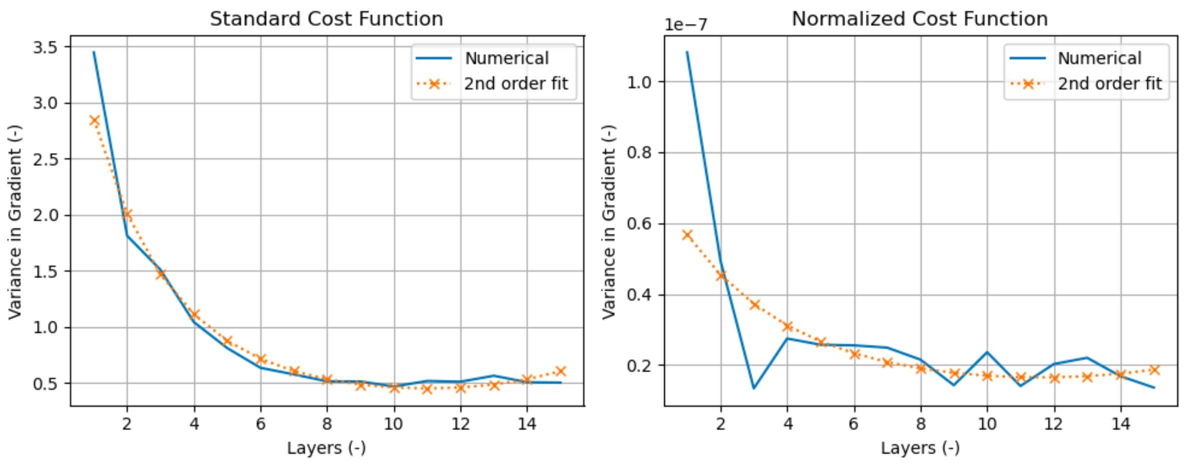


Figure 5.15: Variance versus the number of layers for the standard and normalized cost function. 5 Qubits are used for all layers.

A potential method for obtaining the required amount of layers for an ansatz to achieve a sufficient expressibility, might be by looking at the variance of the gradients in the number of qubits. As the expressibility is increased, it seems that the variance keeps reducing. Until a point is reached where the Hilbert space, which is spanned by the ansatz, is saturated and the variance reaches a lower limit. However, to conclude this more research in this direction should be done.

5.5.1. What Shapes The Cost Function Landscape?

Now that it is known that the normalized cost function suffers from an exponentially vanishing gradient in the number of qubits, it is interesting to investigate why this occurs. By looking at the figures regarding the variance of the gradients in the previous sections, it can be seen that the scaling in the number of qubits is especially detrimental for the trainability of the normalized cost function. But what causes this? The normalized cost function is defined as:

$$C(\theta) = 1 - \frac{|\langle b|A|\psi\rangle|^2}{\langle\psi|A^2|\psi\rangle} \tag{5.1}$$

So the only terms that influence the cost function landscape are $\langle\psi|A^2|\psi\rangle$ and $|\langle b|A|\psi\rangle|^2$. By looking at how these values scale when the number of qubits is changed, it might be possible to conclude why the gradient vanishes.

In fig. 5.16 various terms, obtained with classical methods, relevant to the cost function are plotted versus the number of qubits. The plot on the left shows how the term $\langle x|A^2|x\rangle$ (and $|\langle b|A|x\rangle|^2$) scales. Here it immediately becomes clear why the vanishing gradient occurs. As the problem size is increased, both terms in the cost function become exponentially smaller in size. On the y-axis only $\langle x|A^2|x\rangle$ is mentioned, but since the cost function only equals zero when $\langle x|A^2|x\rangle = |\langle b|A|x\rangle|^2$, the same values also hold for the $|\langle b|A|x\rangle|^2$ term at the solution.

The slope of the first order approximation results in -2.68, which is close to the value of -2.82 as was found in fig. 5.12. It should be noted that the value of -2.82 was obtained by doing 200 random initialization of the ansatz parameters and can thus change depending on the different runs. On the right hand figure various other terms are plotted. These are the $\lambda = \frac{1}{\langle b|A|x\rangle}$ term used for the scaling of the normalized solution. xA^2x corresponds to the classical equivalent of $\langle x|A^2|x\rangle$ to indicate how solving the classical solution scales. The term $|b|$, which is the normalization factor of the right-hand side vector and finally the condition number κ of the A-matrix is shown. Here an important point is that, for the classical term xA^2x, the A-matrix with the $1/h^2$ scaling is used. While for all quantum terms, such as in $\langle b|A|\psi\rangle$, it is defined without the $1/h^2$ scaling. This is done since in the quantum implementation the scaling does not affect the trainability of the ansatz, thus does not have any added benefit of adding to the terms.

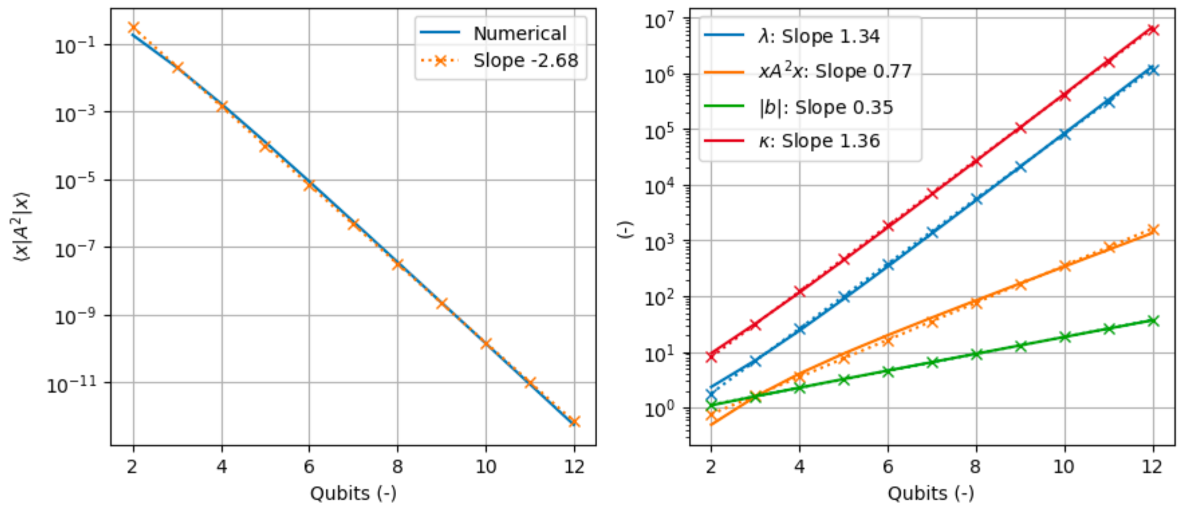


Figure 5.16: Scaling of various terms versus qubit size. In the left figure the scaling of the cost function term $\langle x|A^2|x\rangle$ is shown. While on the right other terms such as λ, the classical counterpart xA^2x, the normalization factor $|b|$ and the condition number κ are shown.

In the figure on the right it is shown that the slope of the condition number is 1.36. However, the slope for $\langle\psi|A^2|\psi\rangle$ is almost twice as large. This stems from the fact that the matrix occurs squared in this term and has the effect that the slope on the log scale doubles. For the standard matrix A it holds that $Ax = \lambda x$, where λ here refers to an eigenvalue and not the scaling term. For a squared matrix it holds that:

$$A^2x = A(Ax) = A\lambda x = \lambda(Ax) = \lambda^2 x \tag{5.2}$$

Meaning that the terms in the condition number are squared for the A^2 matrix as compared to the A matrix. As a result, the slope for the condition number of A^2 is twice that of A and results in 2.72. This number is very close in size to that of the -2.68 found for the $\langle x|A^2|x\rangle$ term and the difference can be attributed to the fact that $|x\rangle$ is a normalized vector. For a normalized vector, the square of the individual terms sums to one, meaning that the sum of the non-squared terms actually is larger than one when not squared. As the vector gets larger, this effect becomes stronger and the steepness of the slope is slightly reduced. The slope for κ is positive on the right and negative for $\langle x|A^2|x\rangle$ on the left due to the fact that for the A matrix without the $1/h^2$ term, the smallest eigenvalue keeps decreasing.
This troublesome scaling of $\langle x|A^2|x\rangle$ also explains why in the quantum case the accuracy is more of a problem than in the exact linear algebra case. For each increase in problem size, the required accuracy to accurately capture the expectation value of the terms in the cost function, increases by roughly a factor of $e^{2.8} \approx 16$. Leaving hardware noise outside of this discussion, it is known that the relation between the error ϵ and the number of shots is inversely related by $\epsilon \propto \frac{1}{\sqrt{N}}$, where N is the number of shots. This means that to reach an accuracy in the same order as the terms in the cost function corresponding to the solution, a number of shots proportional to $N \propto \frac{1}{\epsilon^2} = \frac{1}{\langle x|A^2|x\rangle^2}$ is required. This relation is shown in fig. 5.17 and makes it very clear that it becomes effectively impossible to solve the Poisson equation with this implementation of the VQLS algorithm for problems larger than 4 qubits when shot noise is included. Since the requirement for the number of shots is too large to evaluate in a reasonable amount of time.

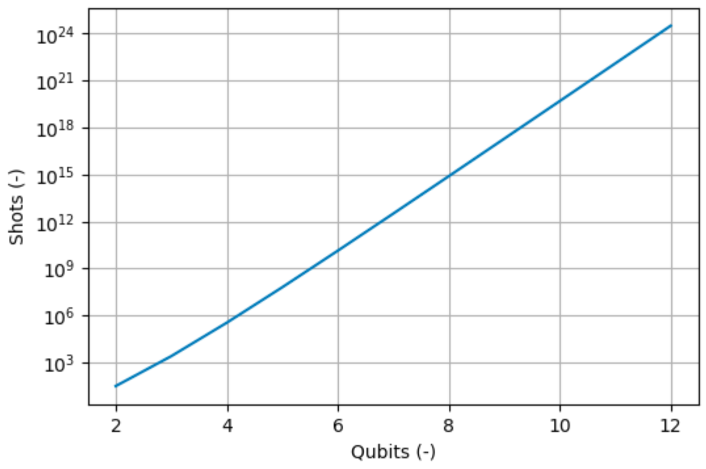


Figure 5.17: Number of shots required to reach an accuracy in the order of the terms in the cost function at the solution versus the number of qubits.

5.5.2. Scaling of the A-matrix

As mentioned in the previous sub-section, classically one solves for the matrix $A^* = A/h^2$. Since on a quantum computer only normalized states are used, rescaling of A does not make any difference in the approach. In the end, one is only evaluating an expectation value in the form of $\langle\psi|A_i|\psi\rangle$, where A_i is a normal operator. For example in the normalized cost function it becomes clear immediately why this does not have an effect:

$$C_{norm}(\theta) = 1 - \frac{|\langle b|A^*|\psi\rangle|^2}{\langle\psi|{A^*}^2|\psi\rangle} = 1 - \frac{|\langle b|A/h^2|\psi\rangle|^2}{\langle\psi|A^2/h^4|\psi\rangle} = 1 - \frac{h^4}{h^4}\frac{|\langle b|A|\psi\rangle|^2}{\langle\psi|A^2|\psi\rangle} = 1 - \frac{|\langle b|A|\psi\rangle|^2}{\langle\psi|A^2|\psi\rangle} \tag{5.3}$$

Thus rescaling of the matrix, classically does have an effect on the eigenvalues, however on a quantum computer this is not the case. That means that it also does not influence the cost function landscape and will not help with improving the trainability. The individual terms are scaled in size as the factor of $1/h^2$ is added, however, this only has the effect of changing the scale of the cost-axis in a cost function landscape and does not actually change the shape of the landscape itself. Thus also for the standard cost function this does not make an effect on the ansatz trainability.

5.5.3. Convergence of Cost Function Terms

Further investigation of the convergence of the terms $\langle\psi|A^2|\psi\rangle$ and $|\langle b|A|\psi\rangle|^2$ can also lead to greater insights in why the optimization is difficult. When the cost function is equal to zero, it must be that $\langle\psi|A^2|\psi\rangle = |\langle b|A|\psi\rangle|^2$, for both the standard and normalized case. During an exact state vector optimization this cost function is always valued larger than zero, meaning that during optimization $\langle\psi|A^2|\psi\rangle > |\langle b|A|\psi\rangle|^2$ holds. Since for the normalized cost function the landscape is mostly flat for larger problem sizes, this indicates that $\langle\psi|A^2|\psi\rangle \gg |\langle b|A|\psi\rangle|^2$. By running the optimizer for a certain problem and keeping track of the individual terms during optimization at each iteration, the progression of the terms is followed.

A general example of this is given in fig. 5.18 and fig. 5.19. Here results for a 3-qubit problem with 3 layers of the HEA are shown. Results are from the exact method with the SLSQP optimizer. Both figures show results from the same optimization procedure. On the left the individual terms $\langle\psi|A^2|\psi\rangle$ and $|\langle b|A|\psi\rangle|^2$ are tracked versus the $L2$-norm with the error and the fidelity with the final solution vector $|x\rangle$. The $L2$-norm is taken of the error between the quantum state $|\psi\rangle$ and the normalized exact solution, which was computed beforehand. The fidelity is defined as $\langle x|\psi\rangle$ and indicates the amount of overlap between the quantum state and the solution. On the right the value for the standard and normalized cost functions are shown with the $L2$-norm and the fidelity.

In the figure, it becomes clear that only when $\langle\psi|A^2|\psi\rangle$ and $|\langle b|A|\psi\rangle|^2$ are very close together, the $L2$-norm of the error starts to reduce significantly. After around 2500 iterations the fidelity starts to get very close to 1 and around this mark the $L2$-norm with the error starts to reduce significantly. At around the same iteration, the value of the normalized cost function is still relatively high at around $C = 0.8$. While the standard cost function is already close to two orders of magnitude smaller at around $C = 0.01$. From these figures it is also clear that the statement about $\langle\psi|A^2|\psi\rangle > |\langle b|A|\psi\rangle|^2$ indeed is correct.

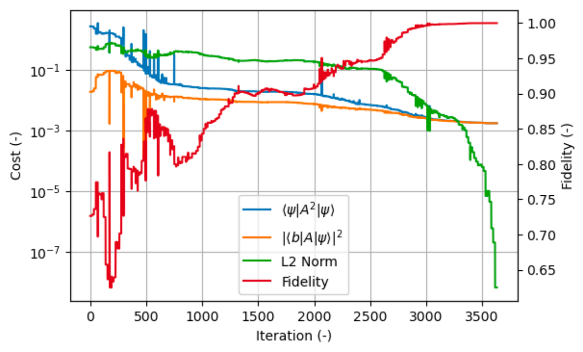


Figure 5.18: Convergence of the cost function terms for a 3-qubit problem.

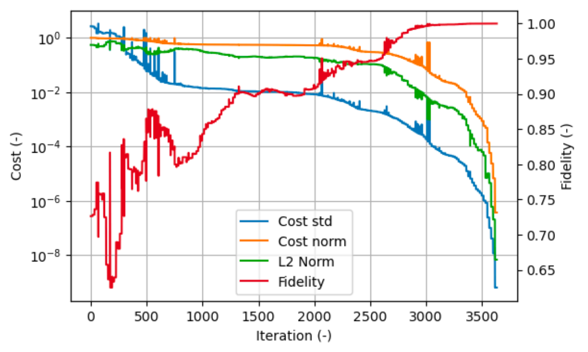


Figure 5.19: Convergence of the standard and normalized cost function for a 3-qubit problem.

In fig. 5.20, the values of the terms are plotted during the optimization process of a 4-qubit problem. The results shown here are obtained with an exact simulation where no shot noise is taken into account. Again, only when the two terms $\langle\psi|A^2|\psi\rangle$ and $|\langle b|A|\psi\rangle|^2$ are very close together the $L2$-norm starts to reduce. Regarding the fidelity, it looks like after around 1500 iterations, it reaches a high value of above $F = 0.95$. However, if one would plot the vector of the quantum state at that iteration and would compare it to the true solution, there would still be significant differences. Another difficulty that this figure shows, is that the optimizer runs for too long. Only considering the fidelity, optimization could have been stopped at around 2500 iterations. More than half of the iterations are after this point, which only marginally improves the final fidelity. After around 4000 iterations the $L2$-norm starts to quickly drop, which can also indicate that the optimization is at a sufficient level. However, neither the fidelity nor the $L2$-norm is possible to efficiently extract during the optimization procedure. So the only reliable option is to wait until the optimizer has reached a sufficiently low cost function value, which can be difficult to judge as different problem sizes have different fidelities at the same cost value.

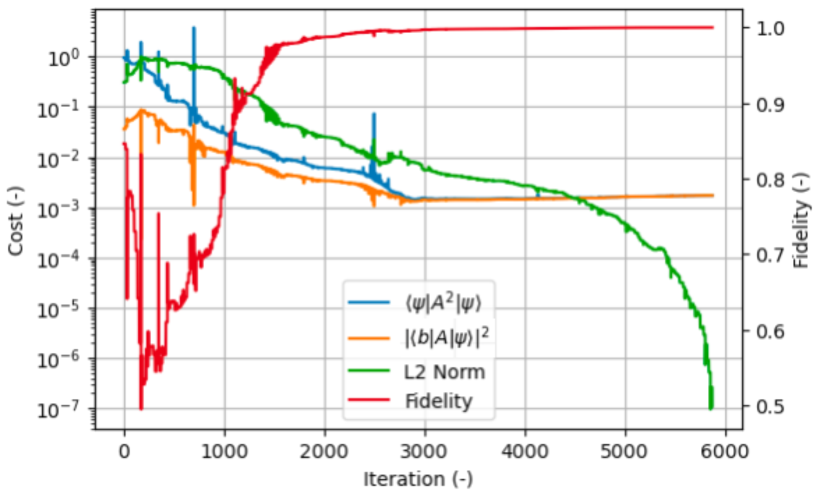


Figure 5.20: Convergence of the terms for a 4-qubit problem.

5.6. Shot Noise and Accuracy

Including shot noise during the optimization procedure results in another difficulty. As shown in previous sections the absolute value of the terms $\langle\psi|A^2|\psi\rangle$ and $|\langle b|A|x\rangle|^2$ reduce exponentially in the number of qubits. Meaning that an increasing number of shots is required to achieve a sufficient accuracy. However, there is an additional difficulty in finding the number of shots required to achieve a certain accuracy. This is due to the decomposition in the form of $A = \sum_i c_i A_i$. During the computation all the individually evaluated terms will be multiplied by their respective coefficient and added to the total. This means that any error in the computation of A_i, will be multiplied by c_i.

In the decomposition of A this is not that big of a problem, since here all the coefficients are either plus or minus one, apart from the $I^{\otimes n}$ operator which is multiplied by 2. The decomposition of the A^2 matrix contains coefficients with values of -4. This leads to the fact that the errors in these terms is also multiplied by 4. After adding and subtracting all the individual terms, multiplied with their respective coefficients, this means that some terms have much larger errors than others. Even though the relative errors of the individual terms can be very small (order of 0.1%), the total error of the combined terms can be several orders of magnitude larger.

An example for the 2-qubit case is shown in table 5.1. Here the operator indicates A_i, the coefficient indicates the c_i terms, E_{exact} indicates the exact evaluation of $\langle x|A_i^2|x\rangle$, E_{noisy} a noisy evaluation of $\langle x|A_i^2|x\rangle$ with 10,000 shots, the absolute error is determined by $|E_{exact}$-$E_{noisy}|$, while the relative error is computed by eq. (5.4). Small differences in the absolute error and relative error can be present due to the rounding applied in the table. The results shown here are of course randomly obtained and each different evaluation of 10,000 shots for all the terms in the decomposition will yield different outcomes.

$$\text{Relative Error} = 100\left|\cdot\left(1-\frac{E_{noisy}}{E_{exact}}\right)\right| \tag{5.4}$$

In the table it can be seen that the largest total error is 0.011 for the $I\sigma_-$ term and the largest relative error occurs for the 0-state and is 2.65%. The overall error is 0.019, which is similarly to the largest error of the individual terms. However, the relative error of the final value is 10.28%. Which is around 4 times as large as the largest individual error. This is primarily due to the multiplication with the coefficient, which increases the error. This relative error becomes a larger problem for larger qubit sizes as the absolute value of the $\langle x|A^2|x\rangle$ and $|\langle b|A|x\rangle|^2$ terms goes down. For the 2-qubit case this value is $\langle x|A^2|x\rangle = 0.181818....$, and having an uncertainty of ±0.01 does not make a significant difference in the total value. However, for the 5-qubit case this value is $\langle x|A^2|x\rangle = 0.00013$ and having an uncertainty of ±0.01 in this case yields the evaluation completely useless. As this time the error is several orders of magnitude larger then the expectation value itself.

Table 5.1: Evaluation of terms in the decomposition of A^2 for 2-qubits. The exact value, value with shot noise obtained after 10,000 shots, absolute error and relative error are shown.

Operators	II	$I\sigma_+$	$I\sigma_-$	$\sigma_+ I$	$\sigma_- I$	$\sigma_+\sigma_-$	$\sigma_-\sigma_+$	0-state	1-state	Total
Coefficient	6	-4	-4	1	-4	1	-4	-1	-1	-
E_{exact}	1	0.461	0.461	0.448	0.448	0.339	0.339	0.097	0.218	0.182
E_{noisy}	1	0.454	0.472	0.456	0.440	0.336	0.335	0.094	0.218	0.201
Absolute Error	0	0.007	0.011	0.008	0.008	0.003	0.004	0.003	0.001	0.019
Relative Error	0	1.50	2.41	1.56	1.87	1.03	1.44	2.65	0.31	10.28

The $|\langle b|A|\psi\rangle|^2$ terms have smaller coefficients and thus the total error in the evaluation is on average smaller than that of $\langle\psi|A^2|\psi\rangle$. After all, it still holds that if the number of shots is increased, the error will be reduced. However, the effect as described above leads to the fact that a larger number of shots is required to reach a certain accuracy as what would be initially expected.

Another aspect where this becomes a problem, is during the evaluation of the cost function close to the solution. The final value of the terms in the cost function is a very small number and if the uncertainty due to an insufficient number of shots is as large as the value itself, negative numbers can result as an output. In the case for the normalized cost function this means that if the $\langle\psi|A^2|\psi\rangle$ term becomes slightly smaller than zero, the cost value itself can quickly increase in value and blow up in size. This is due to the fact the term is in the denominator and quickly causes the total fraction to become larger. At the beginning of the optimization procedure the effect of the relative error is of less importance, but once one it is getting closer and closer to the true solution, inaccuracies in the evaluation of the expectation values can cause random spikes in the cost function landscape. When far away from the solution, this is less of a problem, since then the absolute values of the two terms are larger and a small error has a less significant influence.

To demonstrate this effect the $\langle x|A^2|x\rangle$ term is evaluated at the solution for the 2 and 3 qubit case with the noisy quantum implementation. This is repeated 1,000 times for 3 different numbers of shots. Then for each case a curve fit is constructed and shown in fig. 5.21 for the 2-qubit case and fig. 5.22 for the 3-qubit case. For the 2-qubit case the optimized value for $\langle x|A^2|x\rangle = 0.1818$. It is clear that when only 100 shots are used, a significant portion of the tests resulted in a negative value. When 1000 shots are used, this number is already much lower and for the 10,000 shots case it is extremely unlikely to find a negative term. This also indicates why the 2-qubit case can be optimized with only 10,000 shots. For the 3-qubit case on the right it is a different story. The optimized value for $\langle x|A^2|x\rangle = 0.020$, which lies much closer to zero. For the cases with 10^2, 10^3 and 10^4 shots the number of negative outputs is relatively large. Only when more than 10^5 shots are used, is the chance of obtaining a negative value around 2 standard deviations away from the mean.

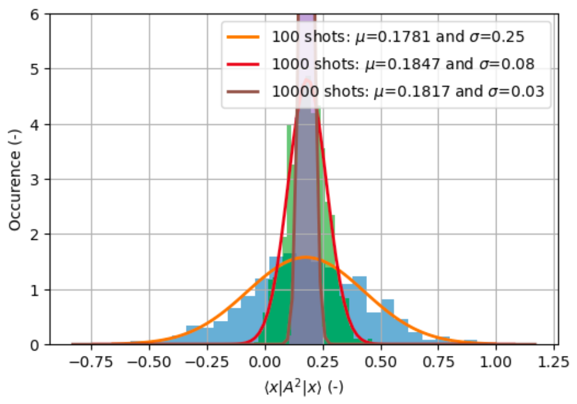


Figure 5.21: Histogram with curve fit for a 2-qubit problem evaluated 1000 times for different shot numbers.

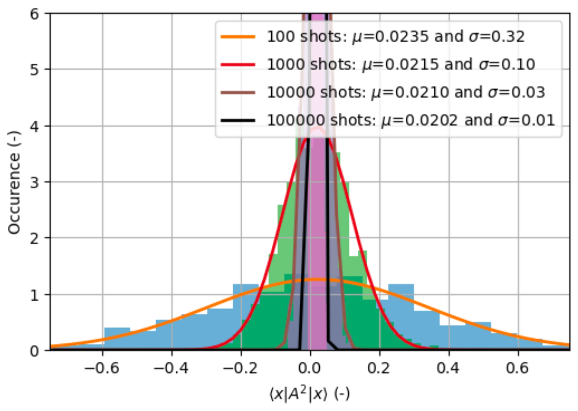


Figure 5.22: Histogram with curve fit for a 3-qubit problem evaluated 1000 times for different shot numbers.

5.7. Extracting Quantum Information

As mentioned in section 4.8, it might be possible to efficiently extract classical information out of the quantum solution in order to reconstruct the solution vector. It should be noted that this only works if the shape of the solution is known beforehand and a polynomial fit of the correct order is made through the known points. The standard optimization procedure as mentioned in the linear algebra approach will be used to obtain the results. In order to extract a value close to the maximum value of the solution, the $x = 0.6$ position is used. This is found by determining the location of the quantum state corresponding closest to $x = 0.6$. For the 2-qubit case this is state 3 [1], for the 3-qubit case state 4 is used, for the 5-qubit problem state 19 is used. Figure 5.23 shows the error between the estimated maximum value of the solution obtained from the Swap-test with 100,000 shots and the classically computed maximum value. As can be seen errors well below 1% are achieved. An example of the 5-qubit case is shown in fig. 5.24. Here the exact solution is compared to the third order polynomial fit. The $L2$-norm between these two states results in a value of 0.0045. For all problem sizes below 5-qubits the $L2$-norm of the error is below 10^{-2}, indicating that this approach indeed works well for approximating the solution.

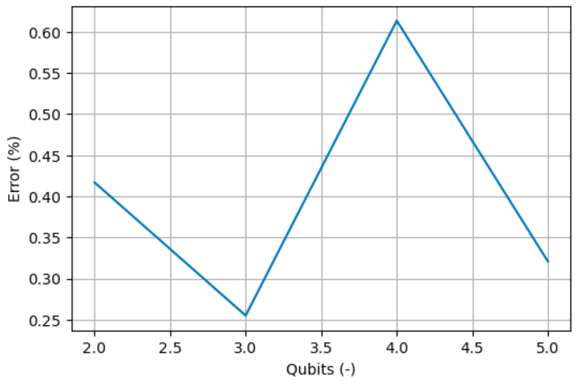


Figure 5.23: Error in the estimated maximum amplitude obtained from the Swap-test with 10^5 shots.

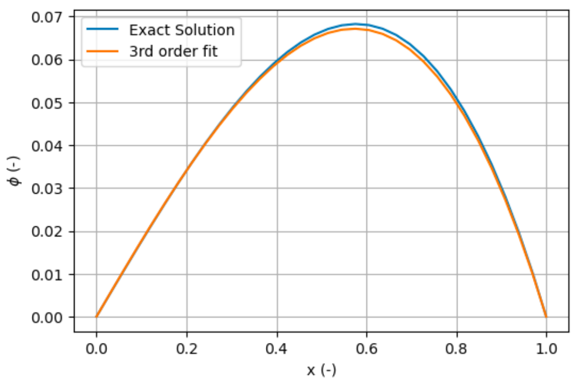


Figure 5.24: Solution shown for a 5-qubit problem

Another way of extracting useful information is by using the reference vector $v_{ref} = |b\rangle$. This way one measures the overlap $\langle b|x\rangle$, which can be regarded as a measure for the energy in the system. This is a measure of the fidelity between the two states and is also evaluated by using the Swap-test. As the problem size is increased this value for the energy should converge to a certain value. Figure 5.25 shows how the term $\langle b|x\rangle$ converges as the number of qubits is increased. Here it is clearly seen that the value converges to around $\langle b|x\rangle \approx 0.838$.

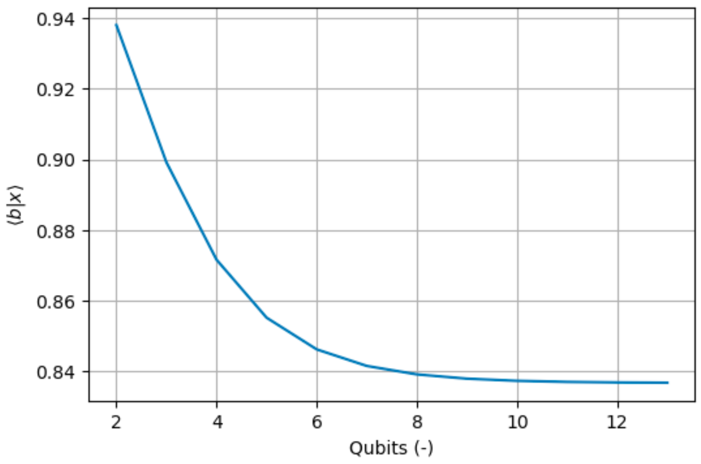


Figure 5.25: Fidelity $\langle b|x\rangle$ for different problem sizes.

[1] Here the third state refers to the third position. For a 2 qubit problem this results in the vector (0,0,1,0).

5.8. Adapted Poisson Equation

Now that it is known that the standard Poisson equation is indeed quite difficult to solve, the adapted form of the Poisson equation might give better results. First the scaling of the cost function terms is shown in fig. 5.26 for different cases of $-\nabla^2\phi + c\phi = x$, where c is being varied. Here $c = 0$ corresponds to the original problem, for which $\langle x|A^2|x\rangle$ exponentially decreases in problem size. When c is increased, the exponentially decreasing behaviour is not present anymore and starts to approach a limit of c^2. This means that, at least in theory, no exponentially increasing amount of accuracy is required to obtain the solution.

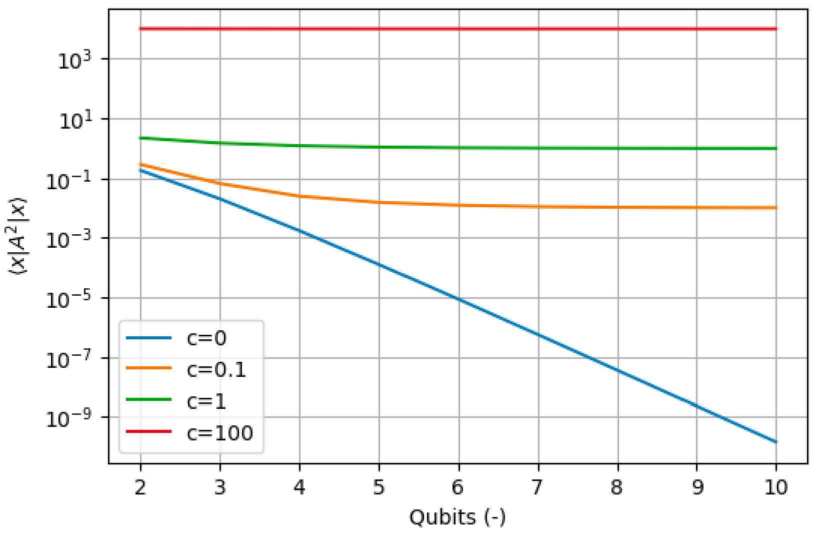


Figure 5.26: Scaling of the $\langle x|A^2|x\rangle$ term for the adapted Poisson equation with different values of c.

By constructing the cost function landscape it can be verified whether this actually makes the problem better solvable. Plotting the landscape for various different coefficients will allow for easy visual comparison. Rather than a 3D surface plot now a 2D contour plot is used, to better indicate the changes between the different functions. In fig. 5.27 the cost function landscape is shown for 6 different coefficients, which are shown in the title of the subplots. These range from a very extreme case of $c = 100$ back to $c = 0$. What was expected is indeed the case, when the coefficient is large the gradients in the landscape are also large.

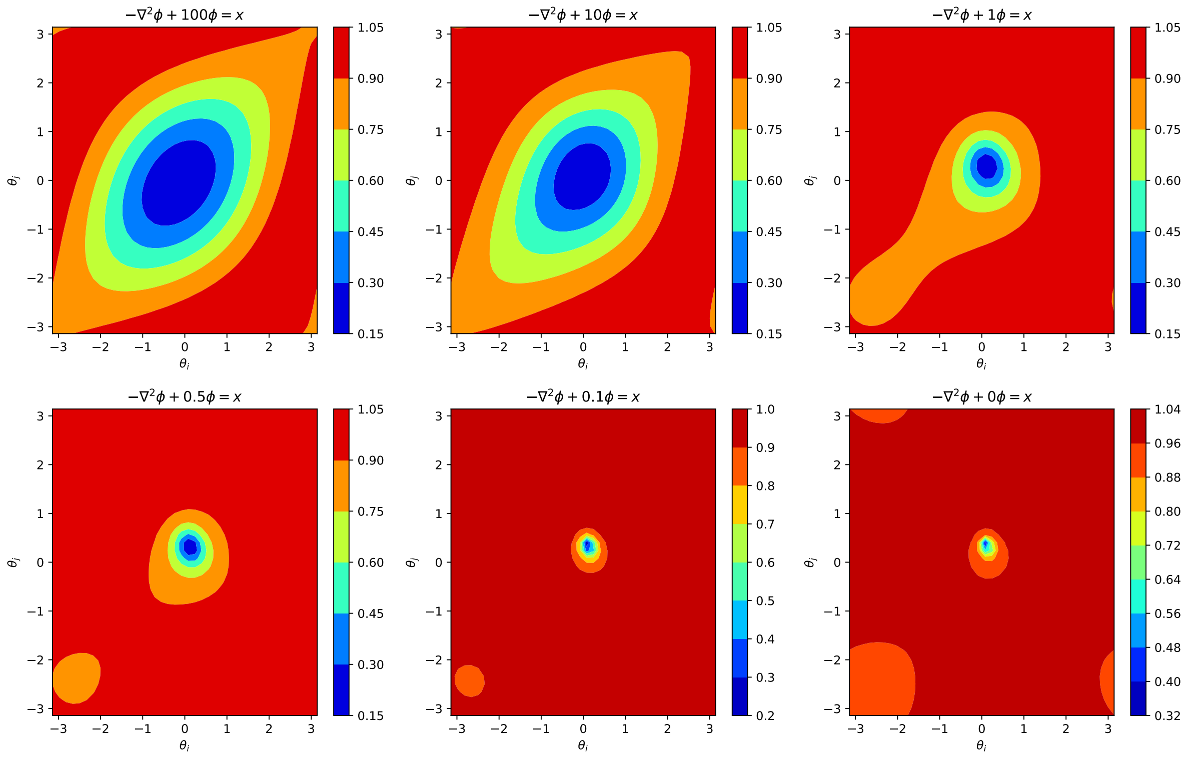


Figure 5.27: 2 Dimensional view of the normalized cost function landscape for 6 different values of c in $-\nabla^2\phi + c\phi = x$.

By just looking at the above figure, one could quickly conclude that the solution of $c = 0$ and that of $c = 100$ lie extremely close together, and it should be easy to move between the two. However, this is not the case. There is actually a difference between them, which is shown in fig. 5.28. As the coefficient becomes larger, the parabolic solution curve shifts more and more to the right and approximates a linearly increasing curve, excluding the boundary conditions. This means that one is actually obtaining a slightly different solution when using the adapted form. It is possible to first optimize for a larger value of c and then slowly reduce it towards $c = 0$ in order to obtain the solution to the original problem. However, this is again a non-trivial problem to solve. Reduce c too quickly during optimization and the benefit of larger gradients cannot be fully used. Reduce c too slowly and the optimizer will think it is at a minimum, or the process will take a relatively long time.

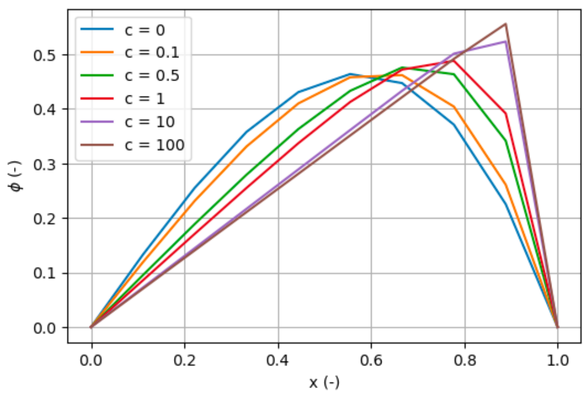


Figure 5.28: Solution of the adapted Poisson equation for 6 different coefficients.

The same figure, as was made for the normalized cost function, can also be made for the standard cost function. This one is shown in fig. 5.29. Now only $c = 1$ and $c = 0$ are shown, since the contour plots themselves do not differ so much. The only notable difference is the upper limit on the color bar next to the figure. In the case of $c = 1$, the maximum is around 20, while for the $c = 0$ case it is only around 12, clearly indicating that the overall gradients are larger.

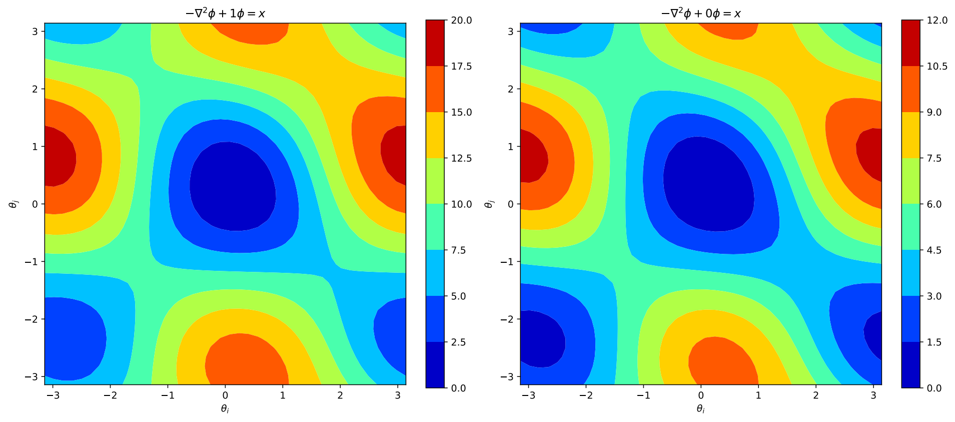


Figure 5.29: 2 Dimensional view of the standard cost function landscape for $c = 1$ and $c = 0$ in $-\nabla^2\phi + c\phi = x$.

The approach of using the adapted Poisson equation might seem similar to rescaling the A-matrix as shown in eq. (5.3), such that the same scalar as in classical methods is obtained, which did not work. However, the main difference here is that the previous approach relied only on scaling the complete A-matrix, which basically corresponds to evaluating the expectation values of the individual terms $\langle\psi|A_i|\psi\rangle$ and then multiplying with their respective coefficient. Here the limiting factor is that the error which stems from the initial expectation value also scales with the multiplication factor in front of it. So making this number much larger or smaller does not have any noticeable effect, as the error scales with it as well. In the case for the adapted Poisson equation , the A matrix itself is actually changed and thus becomes easier to solve for a different reason.

5.8.1. Morphing of Poisson Problem

One method to still be able to obtain the original solution to the Poisson equation, is by slowly morphing the adapted equation back to the standard Poisson equation. This can be done by varying c based on a predetermined number of iterations or cost function values. Another method would be to vary c based on a function depending on the iteration number. However, both of these approaches suffer from the fact that each different problem size would require a different speed at which c is varied. For smaller problems, the gradients are larger and c must be varied more quickly. For larger problems, the gradients are smaller and c must be varied more slowly.

The main difficulty is that this approach only works with a correct reduction in c during the optimization procedure. If c is reduced too quickly, the optimizer will not keep track of the local minima it is in at the moment and it will effectively only solve the original Poisson equation, where $c = 0$. If c is reduced too slowly, the optimizer might think that it is at the true global minimum and stop the procedure. This can be prevented from happening, by only forcing the optimizer to stop when $c = 0$, but this can lead to a very large number of iterations, meaning that the optimization procedure takes an unnecessarily long amount of time. After testing various method with changing c based on: iteration number, cost value, number of function evaluations and gradient values, no solid approach was found where optimization could be improved over just using the standard or normalized cost function.

5.8.2. Morphing of Adapted Problem Cost Function

The problem of manually changing c can be avoided by added c as an extra parameter for the optimizer. This way the optimizer itself varies the coefficient. By introducing a cost function in the form of eq. (5.5) it is guaranteed that the minimum is obtained only when $c = 0$. In this process c is also used as the coefficient for the adapted Poisson equation as in $-\nabla^2\phi + c\cdot\phi = x$. One restriction for this cost function is that a lower bound of $c = 0$ must be set. Otherwise the optimizer will immediately try and make c as negative as possible in order to minimize the overall cost function. By setting a lower bound of $c = 0$ for the optimizer on this specific parameter, this is prevented.

$$C(\theta) = 1 - \frac{\langle b|A|\psi\rangle|^2}{\langle\psi|A^2|\psi\rangle} + c \tag{5.5}$$

This cost function is tested between 2 to 6 qubit problems with the linear algebra method. For the 2 and 3 qubit cases optimization always succeeded. An example for the 4-qubit case is shown in fig. 5.30. Here the value of c, the cost as defined in eq. (5.5) and a result with the normalized cost function are shown. The results shown in the figure for the two cost functions are using the same initial ansatz parameters. It is clear that this adapted form gets to a lower cost value earlier than the normalized cost function. However, in total a very similar amount of iterations is needed before convergence is reached. Generally the 4-qubit problems where slightly faster with this adapted form of the normalized cost function than the original form. For the 5-qubit problems generally the same performance was found. However, sometimes the optimizer failed to find the true solution for $c = 0$. An example of this is shown in fig. 5.31. Here a 5-qubit case is shown and eventhough the cost is significantly reduced, near the end the optimizer has trouble finding the true minimum for $c = 0$ and the coefficient stays at a non-zero value.

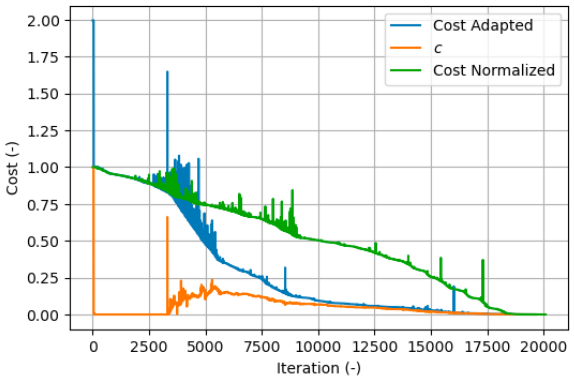


Figure 5.30: Cost function values and c values versus iteration number shown for a successful optimization procedure with eq. (5.5).

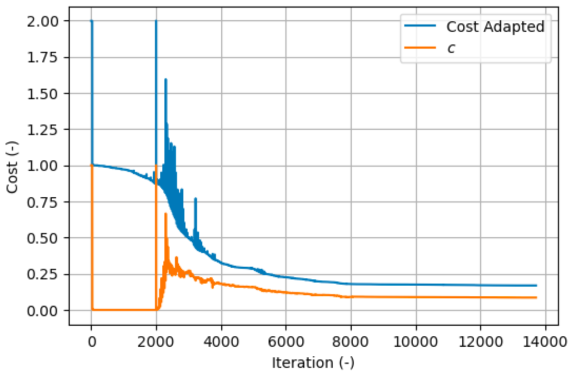


Figure 5.31: Results shown for a successful optimization procedure with eq. (5.5).

During tests with a 6-qubit problem, no successful results were obtained. What happens in most cases is that the optimizer realises it can reduce the coefficient $c = 1$ to $c = 0$ immediately and as a results the gradients become significantly smaller, as it is now solving the original problem. One could prevent this by first setting a limit on the value of c, but then similar problems for reducing the coefficient as in previous sections result. Another problem with this approach is that apart from optimization of the ansatz parameters, the optimizer now also has to solve for c, which affects the complete landscape. So any change in c will result in different optimal values for the ansatz parameters. When c becomes closer to zero, again the gradients become smaller and smaller and often times the optimizer stops before $C = 0$ is reached. Similarly to the previous section, this approach was not useful in improving ansatz trainability and obtaining any improvement over the standard method.

5.9. Combination of Cost Functions

Since both the standard and normalized cost function itself have advantages and disadvantages, a combination of them might prove to be a good result. The standard cost function initially has large gradients, which become smaller and smaller is one gets closer to the solution. For the normalized cost function this is the other way around, where only close to the solution the gradient is large. The following sub-sections show various approaches for different types of cost functions.

5.9.1. Morphing Between Cost Functions

Morphing of the cost functions during the optimization procedure can be done by adding an additional coefficient and using a linear combination of the standard and normalized cost function, as shown in eq. (5.6). Here the coefficient c is used to go from one type of cost function to another. In order to initially benefit, from the properties of the standard landscape and near the end of the optimization procedure from the normalized landscape, c should be decreased from 1 to 0.

$$C_{total} = c \cdot C_{standard} + (1 - c) \cdot C_{normalized} \tag{5.6}$$

An example of what this would look like for a cost function landscape is shown in fig. 5.32. Here the landscape is shown for a 3-qubit problem. By initially making use of the high gradients of the standard cost function, it might be possible to get close enough to the well as found in the normalized cost function.

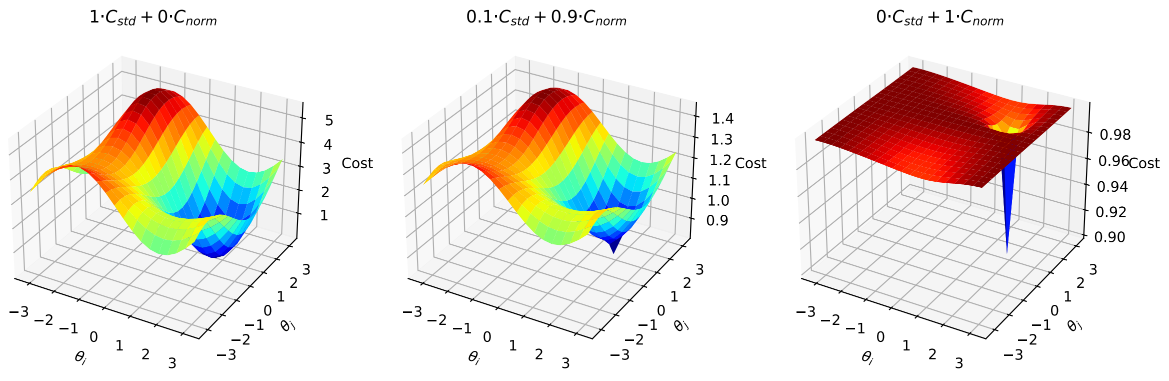


Figure 5.32: Morphing of the cost function landscape for 3 different coefficients.

In the scenario where the cost function values are reduced during morphing, it must happen at such a rate that the minimum is not being lost, which means that one would have to keep track of the rate at which c is changed. As mentioned in the previous section, this is not a trivial task. By adding c as an extra parameter to the optimizer, it is not required to manually change the parameters. Now the cost function is slightly changed as shown in eq. (5.7). Here the c is added at the end, to make sure that $C_{total} = 0$ only holds when $c = 0$ as well. Again c must be bounded by zero such that it cannot attain negative values. This way the optimizer itself will gradually reduce c over time and does not require the need for any manual or function based changes to c.

$$C_{total} = c \cdot C_{standard} + (1 - c) \cdot C_{normalized} + c \tag{5.7}$$

However, with this approach again the same difficulties as with eq. (5.5) were found. For smaller problems the cost function actually converges, but for larger problems the optimizer has troubles finding the true minimum for $c = 0$. This is caused by the same reasons as mentioned in the previous section.

5.9.2. Switching Cost Functions

A slightly different, simpler method which actually seems to be effective is to initially start the optimizer while using the standard cost function and switch to the normalized cost function once a certain cost target has been achieved. This ”switch” cost function benefits from the initial high gradients in the standard cost function and from the high gradients close to the solution in the normalized cost function. Since no morphing takes place during optimization, it is not as difficult for the optimizer to find its way through the higher-dimensional landscape. Initially the standard cost function is used. When a certain threshold for the cost is achieved, the cost function is switched to the normalized cost function. During testing, a threshold of $C_{standard} = 0.01$ is used. This value was chosen after testing for the threshold in the range of $C = 0.5$ to $C = 0.001$ was performed. In this range, $C = 0.01$ generally performed the best and was thus chosen. Again, for larger problems most likely the threshold should be reduced as the size of the well in the normalized cost function reduces as the problem size increases. Thus, the result from the standard cost function should be closer and closer to the true solution as the problem size is increased.

Testing of this method has been done for 2, 3 and 4 qubits with the HEA with 2, 3 and 4 layers respectively. Simulations are done with the exact linear algebra method, in combination with the SLSQP optimizer. Results are shown in table 5.2 for the standard cost function, normalized cost function, the lambda cost function which is explained in the next section and the switch cost function. 50 random ansatz initializations are used to gather the data. So all cost functions are tested with the same 50 random initializations, such that a fair comparison is made. In table 5.2 the average number of cost function evaluations during the 50 optimization procedures are shown. The success rate of the different cost functions is counted as the percentage where the final fidelity of the prepared quantum state $|\psi\rangle$ and the true solution $|x\rangle$ is larger than 0.99. In the table it is shown that the switching cost function has an average number of function evaluations between that of the standard and normalized cost function. Even though the standard cost function has the lowest number of evaluations, for a 4 qubit problem it is shown that the success rate is significantly lower than the other methods. This is due to the fact that local minima can occur and that the gradients near the solution are very small. For the 4-qubit problem, the achieved success rate for the switch cost function is the highest scoring one, while also taking less evaluations than the normalized cost function. This indicates that switching between the two cost functions is a suitable approach and can be used to achieve convergence faster than with just the standard or normalized cost function. Another advantage of this method is that the implementation is not difficult and only requires setting a threshold at which the type of cost function which is evaluated is switched.

5.9.3. Cost Function Lambda Coefficient

Testing with the cost function as shown in eq. (5.8) showed that this cost function indeed works when λ is added to the optimizer as an extra parameter. For 2 and 3 qubit problems λ always converges to $\lambda = \frac{1}{\langle b|A|x\rangle}$. This in itself is an interesting finding, but does not give any additional information, as the term $\langle b|A|x\rangle$ is evaluated during each cost function iteration. Due to the different nature of the cost function it might still give an advantage over the standard or normal cost function.

$$C = \lambda^2\langle x|A^\dagger A|x\rangle - 2\lambda\langle b|A|x\rangle + 1 \tag{5.8}$$

In table 5.2 the average number of cost function evaluations over the 50 runs is shown for each cost function. Here it is clear that the lambda cost function required the most cost function evaluations. However, it also does have a higher success rate than the standard and normalized cost function. This shows that it is a potential alternative to the standard and normalized cost function, but it does require more resources to obtain the same results.

Table 5.2: Average number of function evaluations required for various cost functions. Success rate is determined by percentage of runs achieving a final fidelity with the normalized solution $|x\rangle$ of $F > 0.99$.

	Average Iterations 2 Qubits	Success Rate 2 Qubits (%)	Average Iterations 3 Qubits	Success Rate 3 Qubits (%)	Average Iterations 4 Qubits	Success Rate 4 Qubits (%)
Standard Cost	87.1	100	552.4	100	2805.1	28
Normalized Cost	139.6	100	699.1	100	6052.7	58
Lambda Cost	181.7	100	892.3	100	6400.4	66
Switch Cost	105.1	100	580.2	100	5767.1	72

5.10. Fidelity Cost Function

Using the fidelity as a cost function can be regarded as a type of supervised learning. By using $C(\theta) = 1 - |\langle x|\psi\rangle|^2$, the solution vector $|x\rangle$ has to be known and prepared on a quantum computer prior to running the algorithm. This only makes it useful to implement if one wants to prepare the same state with a different ansatz. Assuming it is known that $U(\theta)|0\rangle = |x\rangle$, then it is possible to use the fidelity to train a different ansatz $V(\theta)$ to also prepare the same $|x\rangle$ state. This is a relatively specific tasks, which does not help for finding the solution to the Poisson equation. However, it can be used to see whether it is possible to train the ansatz when the solution is given. If this generally does not work, then it can also be expected that training the same ansatz with another cost function, which does not directly contain solution information, also will not work.

Testing of this approach is done with the linear algebra method, in combination with the HEA and SLSQP optimizer. To see if the fidelity actually performs well as a cost function, first the cost function landscape is reconstructed. In fig. 5.33, the shape is shown for 2, 4 and 8 qubits. Here the landscape is shaped around the solution to show how the parameters influence the shape. For the 2, 4 and 8 qubit problem, 2, 4 and 30 layers of the HEA are used. Normally the 8-qubit problem would be very challenging to solve, even with the linear algebra approach, as the gradients become extremely small. However, this time with the fidelity as a cost function measure, it is actually possible to obtain meaningful results. In the figure, it immediately becomes clear that the landscape near the solution is independent of the problem size. This makes it seem as if a larger problem would be just as easy to solve as any smaller problem. If the landscape keeps maintaining large gradients, it is possible for the optimizer to find the correct direction towards the solution.

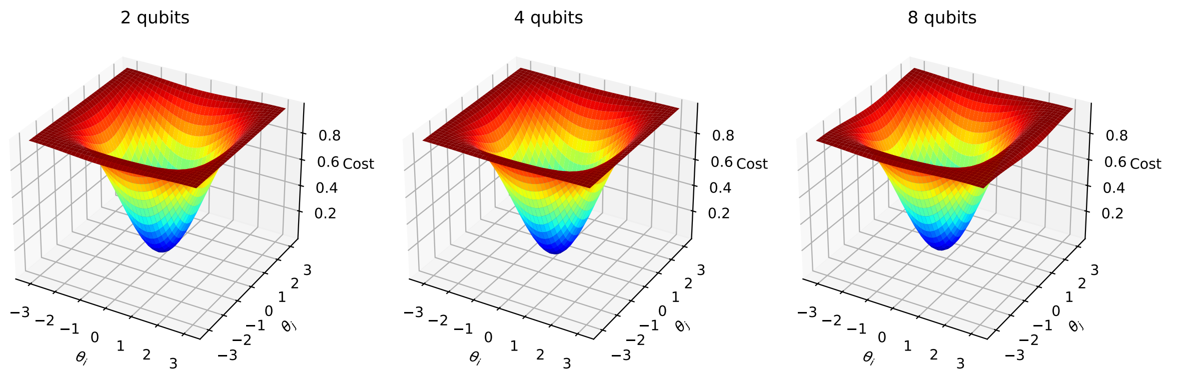


Figure 5.33: Landscape of the fidelity cost function $C = 1 - |\langle x|\psi\rangle|^2$ for 2, 4 and 8 qubits. The center of the landscape corresponds to the solution.

However, when looking at the variance of the gradient for this cost function, a different story is told. In fig. 5.34 it is shown how the variance of the cost function gradient scales for the fidelity cost function. The same process is used as for the previous variance plots. For the normalized cost function a slope of around -2.8 was found, while for the fidelity only a slope of around -1.1 is obtained. This is a significant difference and explains why it is possible to solve larger problems when the fidelity is used as the cost function.

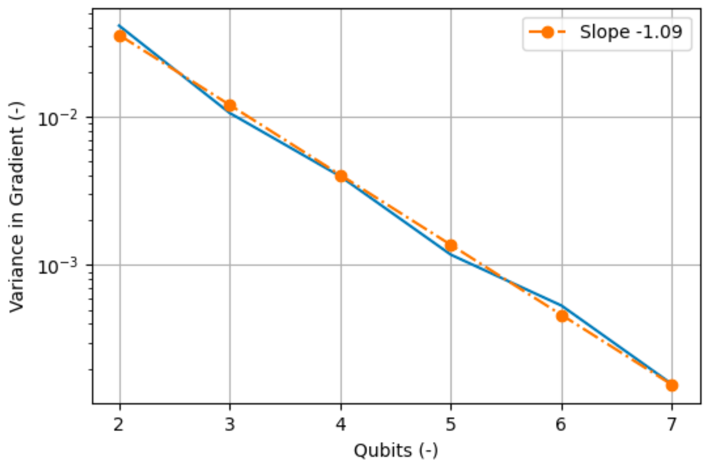


Figure 5.34: Variance of the cost function gradient versus the number of qubits when $C = 1 - |\langle x|\psi\rangle|^2$ is used.

This result does seem contradictory with the results obtained in the landscape figure. How can the landscape stay the same, while the variance of the gradients keeps reducing? This is explained by the fact that, as the problem size is increased, the number of parameters in the ansatz also increases. For the 2 qubit case with 2 layers only $2 \cdot 2 = 4$ parameters are used, while for the 8 qubit case with 30 layers $8 \cdot 30 = 240$ parameters are used. In fig. 5.33 it is seen that, if one is close to the solution **for all parameters**, the landscape is very similar. However, when the parameters are randomly initialized, the chances that some of the parameters are far away from their respective optimized value becomes larger. This in effect reduces the gradient of any other parameter in the ansatz. For example, if any of the 240 parameters in the 8 qubits case is not close to their optimized value, the landscape as in fig. 5.33 will not reach 0, but have a higher minimum. This means that the gradients for the parameters are smaller. So the cause why a decreasing variance is found is that, when more parameters are used, the chance that any of them is far away from the solution increases and that in effect causes a decrease of the gradients for the other parameters.

For each additional qubit, the number of nodes doubles. This means that the slope for the number of nodes equals $ln(2) \approx 0.69$. However, the slope for the variance actually decreases with -1.1, which means that for each additional qubit the variance decreases with a factor of around $e^{-1.1} \approx 0.33$. So the number of nodes gained with an extra qubit does not scale in proportion to the decrease of the variance of the fidelity cost function gradient. Meaning that relatively speaking, one has to spend exponentially more resources when solving larger problems. However, comparing it to the scaling of the condition number of the A-matrix (which was 1.36), the scaling of the variance of the gradients is actually less severe than the degree at which A is approaching singularity.

5.11. Solution Sensitivity

Once the solution state $|x\rangle$ has been found, the problems do not end. So far only the software part of the VQLS algorithm has been discussed, but hardware also plays a huge role when such an algorithm would be implemented on a NISQ device. The output of the classical optimizer is given in a set of parameters which correspond to quantum gate rotation angles and in order to reconstruct the quantum state proportional to the solution vector they must be used in the preselected ansatz. The implementation of these rotation gates is not completely exact due to noise. In order to investigate how significant this inaccuracy is, a short study is done on the influence of rotation noise and shot noise. After the optimized ansatz parameters are found, the quantum state is reconstructed with and without shots noise. In order to simulate noise to the angles in the ansatz, noise data is sampled from a normal distribution with $\mu = 0$ and $\sigma = 0.01$, such that $\theta = \theta_{opt} + \theta_{noise}$, where $\theta_{noise} \sim N(0, 0.01)$. Here the 0.01 value is chosen based on literature. Most literature on the accuracy of gates is specified in fidelity of generating a quantum state and not on the uncertainty on the rotation which is applied to a gate. Currently fidelities in the order of 99.9% are achieved and for simplicity this is translated to an uncertainty of 0.01 degree in the quantum rotation. Having a larger standard deviation in the normal distribution results in larger deviations from the intended rotations in the ansatz, while a smaller standard deviation has the opposite effect.

In fig. 5.35 results for testing of the sensitivity near the solution for 2, 3, 4 and 5-qubit problems are shown. Results are shown for the exact case without noise added to the rotations, the exact case with noise added and 3 cases using noise on the rotations and shot noise. Each data point in the figure is the average of 50 runs with different noise sampling from the normal distribution. As can be seen the exact case without noise has $F \approx 1$ for all problems and $L2$-norm ≈ 0. When noise is introduced the fidelity starts to reduce and the $L2$-norm of the error with the exact solution increases. Various number of shot noises are used to indicate which effect has a larger influence. As the number of shots is increased the limiting factor becomes the noise applied to the rotations of the ansatz. Even though this only has a small influence on the final fidelity, it does increase with problem size and might become a more significant problem when more qubits are used.

5.12. Run Time

Finally, some of the run times taken to evaluate a cost function with the different approaches are analysed. Here the methods between the exact linear algebra approach and the quantum implementation with various numbers of shots will be analysed. In fig. 5.36 the time required to evaluate the normalized cost function a single time is shown versus the number of qubits. The time shown here is only counting the time spent evaluating the cost function itself. All time spent constructing the matrices, vectors and decomposing the A and A^2 matrices is done beforehand and not taken into account. For all cases the HEA is used with 2, 3, 4, 7, 11 and 19 layers for the respective 2 to 7 qubit sized problems. The first plot show the linear algebra approach without using a decomposition. This means that with this approach, after the quantum state $|\psi\rangle$ is obtain from the ansatz a direct multiplication with the matrix A is done, rather than using a decomposition in terms of A_i. The approach with the decomposition is

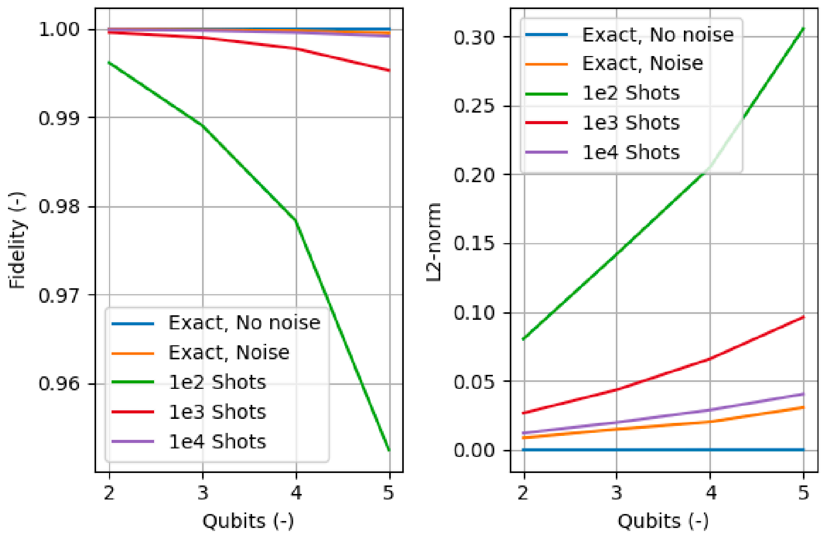


Figure 5.35: Fidelity and $L2$-norm of the error with exact solution shown for exact and noisy methods and different shot numbers. The 2, 3, 4 and 5-qubit problems, use the HEA with 2, 3, 4, and 7 layers respectively.

shown in the plot in the center. On the right, the quantum implementation using different shot numbers is shown. The linear algebra implementations are by far the fastest, with having cost evaluation times in the order of tenths of milliseconds. The linear algebra approach with the decomposition included is taking more time, since now each individual term in the decomposition is evaluated and then only at the end added, instead of computed in one single matrix vector product. For the quantum implementation, the evaluation is taking much more time and takes in the order of seconds for a single cost evaluation. This is mostly due to the generation and execution of all the individual quantum circuits corresponding to the terms in the decomposition. Just by looking at the times required to perform a single cost function evaluation, it becomes clear why the quantum implementation takes so long to evaluate.

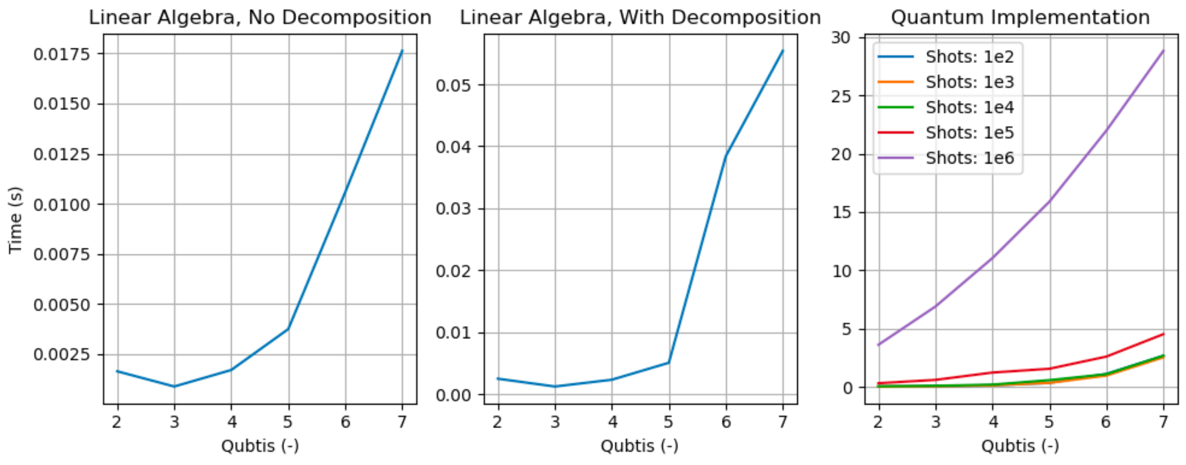


Figure 5.36: Time required for evaluation of a cost function with three different approaches for 2 to 5 qubit problems.

Another interesting aspect is the time required to obtain the gradients when evaluating the PSR with the quantum implementation. These results are shown in fig. 5.37. Here the evaluation of the PSR is plotted versus various number of qubits, for different shot numbers. 2, 3, 4 and 7 layers of the HEA are used for the 2, 3, 4 and 5 qubit-problems. The required time to evaluate is exponential in problem size, as the result is almost a straight line on the log scale.

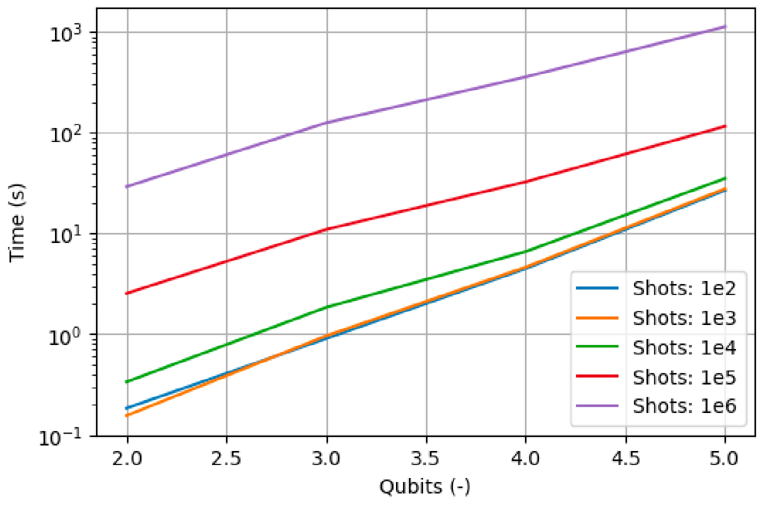


Figure 5.37: Run time for the evaluation of the PSR for different number of shots.

In table 5.3 the data from the figure is shown in a table for more clarity. The results shown here are in seconds. The PSR primarily requires an exponential amount of time due to the increase in the number of parameters. For each parameter the gradient is computed by evaluating the cost function two times, meaning that it is directly related to the number of parameters in the ansatz.

Table 5.3: Time required to compute the full gradient vector by means of PSR with the quantum implementation using various numbers of shots.

Qubits (Parameters) \ Shots	10^2	10^3	10^4	10^5	10^6
2 (4)	0.18	0.16	0.34	2.54	29.17
3 (9)	0.90	0.96	1.84	10.95	125.49
4 (16)	4.47	4.62	6.60	32.55	358.89
5 (35)	26.83	27.79	35.20	116.05	1132.17

6 Conclusions & Recommendations

This chapter covers the conclusions, drawn from the results as shown in the previous chapter. First the general findings are discussed, after which the research question and sub-questions are answered. Finally, a list of practical challenges hindering an efficient VQLS implementation is given.

6.1. General Findings

For smaller problems the VQLS algorithm is capable of finding the solution to the one-dimensional Poisson equation, with Dirichlet boundary conditions and an efficient decomposition of the A-matrix, into raising and lowering operators. However, the trainability of the ansatz quickly becomes a problem when the number of qubits is increased. So it can be concluded that it is indeed possible to solve the Poisson equation by means of VQLS, but whether it brings any benefits over classical methods is another questions. In theory it is possible to also solve problems with 4+ qubits, however this would required an immense number of shots in order to achieve the required accuracy during the evaluation of the cost function terms. Considering the fact that classical approaches are currently several order of magnitude faster, a lot of improvement has to be made before VQLS can compete with classical methods.

The main cause of the difficulties during training is the fact that the cost function terms $\langle x|A^2|x\rangle$ and $|\langle b|A|x\rangle|^2$ become exponentially smaller as the problem size is increased. In order to resolve this, the number of shots required to accurately evaluate the different terms in the decomposition, becomes exponentially large, which affects the total run time of the algorithm. There are methods to improve the trainability of the ansatz. These often require understanding of the problem and tweaking for different problem sizes, which makes them not suitable for general use cases. Examples of these are morphing of the cost function landscape by introducing extra parameters to the optimizer or adapting the original Poisson equation to make the problem better solvable. However, when adapting the original Poisson equation, the problem at hand is changed and should be at some point be morphed back to the original problem.

During the thesis it is found that the very simple linear algebra approach, which only relies on generating the quantum state with an ansatz in Qiskit and then using classical matrix-vector products to evaluate the cost function, brings identical results as the quantum implementation of the algorithm. The advantage of this simplified approach is that it is several orders of magnitude faster than the quantum implementation and can thus save a significant amount of time. This is very useful when new ideas are tested, since the waiting time before results are ready, is not as long.

It should be kept in mind that the results shown here are specific to the ansatze used and an ansatz specifically designed for solving the Poisson equation might provide better results. However, the scaling of the individual terms inside of the cost function are inherent to the problem at hand and thus will not be altered by a different ansatz. So, it is not expected that these problems will completely be solved by a different ansatz. Another very important point to note is that all results shown in this work are obtained using Qiskit, which is an open-source quantum software development kit, which simulates quantum computers. Throughout this whole work it is assumed that Qiskit correctly simulates the quantum processes. However, any discrepancies between the working of quantum hardware in the real world and simulations will not be shown here. Thus the trainability of quantum computers on hardware might behave differently as what is obtained in this work.

6.2. Research Questions

Before the main research questions is answered, first the sub-questions are covered:

Sub-question 1: Does the VQLS algorithm suffer from barren plateaus when an efficient decomposition consisting of raising and lowering operators is used?

Analysis of the results indeed showed barren plateaus for the normalized cost function while using a decomposition in raising and lowering operators. For the standard cost function, these plateaus are not present. However, this cost function still is not capable of obtaining solutions for larger problem sizes, as local minima are present and the gradient becomes very small in the vicinity of the solution. An interesting thing to note, is that identical results are obtained for a quantum implementation and a simplified linear algebra approach. Since the full quantum state is rotated into the Bell basis and the normalized cost function in fact suffers from Barren plateaus. It can be concluded that the decomposition into raising and lowering operators, indeed behaves as a global operator and is affected by barren plateaus. This means that the algorithm, most likely will not be useful to solve larger instances of the Poisson equation in the future, unless a method is found to circumvent the problem of barren plateaus.

Sub-question 2: What causes vanishing of the gradients for certain cost functions?

Investigation of why barren plateaus occur, required an analysis of how the individual terms in the cost function scale in problem size. The scaling of the terms used in the cost function seems to be a problem unavoidable when solving the Poisson equation by means of VQLS and could potentially also be extended to other problems. In the case of the one-dimensional Poisson equation with Dirichlet boundary conditions the terms $\langle\psi|A^2|\psi\rangle$ and $|\langle b|A|\psi\rangle|^2$ vanish exponentially in the number of qubits. Meaning that as one increases the problem size, the number of shots required to achieve a sufficient accuracy, increases exponentially. This has the effect of the algorithm requiring an exponential amount of time to run and makes it, in its current state, slower than its classical competitors.

Sub-question 3: Can changes to the cost functions help improve the trainability of the ansatz?

Adapted forms of the cost function, either directly or indirectly by altering the problem, can bring advantages by improving the trainability of the ansatz. However, these often have to be tweaked specifically for a certain problem size and might still suffer from barren plateaus. One method, which does seem to perform well is one where initially the standard cost function is used and is switched to the normalized cost function after a certain threshold in the cost value is reached. This method does however, require tweaking of the threshold value. Other approaches often worked for smaller problems, but again did not yield good results when the problem size is increased to 5+ qubits. Using the fidelity as a cost as in $C = 1 - |\langle x|\psi\rangle|^2$, does perform significantly better than the standard or normalized cost function. Here the downside is that it requires the solution $|x\rangle$ to already be implemented on a quantum computer, which is generally not known a priori.

Sub-question 4: How can information of the quantum state be efficiently extracted and used classically?

Generally the method to extract quantum information once the solution vector $|x\rangle$ is obtained, is done by evaluating $\langle x|M|x\rangle$, where M is some operator giving useful information about the problem. Finding an operator which actually gives useful information is not a trivial thing. The approach tested in this work, where a third order polynomial is used based on the boundary conditions and solution shape to fit to the data, is an efficient way to classically reconstruct the quantum state. By using the Swap-test to extract quantum information on a single amplitude, crucial data is found with limited resources. However, this approach is problem specific and might coincidentally work very well for this instance. For other problems where the solution is highly irregular, simply fitting a polynomial might not be suitable in order to find a high fidelity solution. Another possibility is using the fidelity $\langle b|x\rangle$ as a type of energy measure for the problem. As the problem size increases, this number converges to a certain value. This shows that it is indeed possible to gain useful information in an efficient manner out of the quantum state $|x\rangle$.

Research question: What are the practical challenges when solving the Poisson equation by means of the Variational Quantum Linear Solver?

For small problems the VQLS algorithm is capable of finding the solution. However, as the number of qubits increases, the dramatic scaling of the required number shots makes it, in its current state, nonsensical to use over classical methods. In the future when problem specific ansatze or new forms of cost functions are discovered, it might become a relevant algorithm, but in its current state there are too many difficulties. Even in an exact state vector simulation, without using shot noise, hardware noise, matrix decompositions and exact reconstructions of the quantum state, it is still extremely challenging and time consuming to solve problems containing 6 or more qubits. The main challenges are the scaling of the cost function terms $\langle\psi|A^2|\psi\rangle$ and $|\langle b|A|\psi\rangle|^2$, which become exponentially smaller as the number of qubits is increased. This does seem as a problem inherent to solving the Poisson equation with normalized vectors and overcoming it thus seems like a very difficult task. Since the effect of scaling is so significant, it does occur regardless of the ansatz used. As a result, another practical challenge is the number of shots required to achieve a sufficient accuracy on the evaluation of the cost function. This scales exponentially in problem size and thus leads to a very slow algorithm. Finally, another practical challenge is that of implementing $|b\rangle$. This state must be prepared by another ansatz such that $U(\theta)|0\rangle = |b\rangle$ for an arbitrary vector b. Finding the correct angles θ is a difficult task to do, as it is not straight forward to train such a state on a quantum machine.

6.3. Recommendations for Future Work

After analysis of solving the Poisson equation with an efficient decomposition, various recommendations for future work can be made. First of al, since it is the most significant effect, the scaling of the terms $\langle\psi|A^2|\psi\rangle$ and $|\langle b|A|\psi\rangle|^2$ should be further investigated. These cause the occurrence of barren plateaus and should be resolved before other parts of the problem are covered. For instance, if one finds an ansatz which is ideal to solve the Poisson equation, then sooner or later it will still be a problem regarding shot noise to evaluate the algorithm with sufficient accuracy due to the scaling of the terms in the cost function. The most probable way to avoid this, is by finding a different cost function, which relies on different terms that do not scale so badly. The influence of other hyperparameters is also not expected to resolve this problem, as those only will help with other parts of the problem and not the scaling of the terms inherent to the used cost functions. The list shown below gives some more ideas and problems relevant to investigate during future work.

6.4. Difficulties of the VQLS Algorithm

Implementing a VQLS algorithm still has various hurdles that have to be overcome. Below a brief list is given.

- **Barren Plateaus**: The main difficulty while solving problems by mean of VQLS, as shown in section 5.4 and section 5.5, is that of barren plateaus. These plateaus make an ansatz close to impossible to train and would require an exponentially increasing number of shots in the problem size in order to achieve a sufficient accuracy.

- **Barren Plateaus can only be avoided with a local cost function in combination with a shallow ansatz**: When the ansatz scales linear/polynomially in problem size, then barren plateaus will still occur, as discussed in section 2.9.3. A simple implementation of the hardware efficient ansatz seems to scale linearly in the number of nodes, meaning that it scales exponentially in the number of qubits, which will run into barren plateaus.

- **Extracting useful information out of the quantum state is challenging**: getting useful information out of VQLS is a difficult task. One could measure the overlap with an inner product by using the Hadamard- or Swap-test, but it can be challenging to extract useful information this way. Reconstructing the state with quantum tomography is an expensive tasks, where the number of shots scales with $\mathcal{O}(1/\epsilon^2)$. Furthermore, extracting vectors with negative numbers cannot be done. This work layed out an approach of efficiently extracting amplitudes at certain locations along the domain and using a measure of the energy as in section 4.8 and section 5.7.

- **Efficient decomposition**: Efficient decomposition of the A matrix is not possible in combination with local cost functions. Section 2.8 showed various forms of decompositions. The number of terms with a Pauli decomposition scales exponentially in the number of qubits. Even though such a method would allow for local cost functions, it rules out potential speedup. Decompositions which use non-Hermitian observables can have a very efficient scaling of $\mathcal{O}(1)$ terms in the number of qubits. But so far all these efficient decomposition rely on global operators and will suffer from barren plateaus. Ideally to solve this problem, a type of local cost functions must be found, which works for global operators.

- **Noise Induced Barren Plateaus**: Shot noise and noise coming from quantum gates also induce barren plateaus, as was covered in section 2.9.4. Especially hardware noise will be a very difficult problem to circumvent. Error mitigation solutions are not working against these types of barren plateaus.

- **Preparation of the $|b\rangle$ State**: In most articles it is assumed that $|b\rangle$ is efficiently prepared by a unitary. However, apart from trivial cases, constructing the unitary is a difficult task for larger vector sizes. Generation of the right-hand side vector as a quantum state is covered in section 3.2.

- **Computing Inner Products**: Using the Hadamard-Overlap test requires the use of controlled unitaries as explained in section 2.3. These are prone to requiring a large amount of non-local gates, which have low fidelity on NISQ hardware. Furthermore, they lead to deep circuits that scale polynomial/exponential in the number of qubits, as shown in section 3.1. Again this most likely results in barren plateaus.

- **Level of entanglement throughout the algorithm must be limited**: The ansatz must have a sufficient level of entanglement in order to allow for the generation of the desired quantum state, however, in doing so one will also induce barren plateaus. Thus an ideal ansatz has enough entanglement to generate the quantum state corresponding to the solution, but a low enough level of entanglement to avoid barren plateaus. In section 2.9 a review on expressibility, entanglement and barren plateaus is covered.

- **Achieving results is challenging even in 'perfect' conditions**: Even when shot noise, hardware noise, decomposition of matrices and exact vector representations of the quantum states are used, the solvable problem sizes are still orders of magnitude smaller than what is achievable classically. This might be an effect of quantum computing still being in its infancy stages, but a lot of progress has to be made before problems of a meaningful size can be solved. The results clearly showed in section 5.2, section 5.3, section 5.4 and section 5.5, that obtaining good results is a challenge even in optimal conditions.

- **Scaling of expectation values in problem size can be a problem inherent to VQLS**: As has been shown in section 5.5.1, the individual terms in the cost function become exponentially smaller in the number of qubits. This might be an inherent problem of solving linear systems by means of VQLS and might be impossible to avoid.

Solving only a single of the above points can be a very difficult task and even solving only one of them, is not sufficient for VQLS to be able to solve problems of meaningful size.

Bibliography

[1] Scott Aaronson. *Read the fine print*. 2015. DOI: `10.1038/nphys3272`.

[2] Dorit Aharonov, Vaughan Jones, and Zeph Landau. "A polynomial quantum algorithm for approximating the jones polynomial". In: *Algorithmica (New York)* 55.3 (2009). ISSN: 01784617. DOI: `10.1007/s00453-008-9168-0`.

[3] Dorit Aharonov et al. *Adiabatic quantum computation is equivalent to standard quantum computation*. 2008. DOI: `10.1137/080734479`.

[4] Tameem Albash, Victor Martin-Mayor, and Itay Hen. "Temperature Scaling Law for Quantum Annealing Optimizers". In: *Physical Review Letters* 119.11 (2017). ISSN: 10797114. DOI: `10.1103/PhysRevLett.119.110502`.

[5] Abhinav Anand et al. "A Quantum Computing View on Unitary Coupled Cluster Theory". In: *Chemical Society Reviews* (Sept. 2021). DOI: `10.1039/D1CS00932J`. URL: `http://arxiv.org/abs/2109.15176%20http://dx.doi.org/10.1039/D1CS00932J`.

[6] Andrew Arrasmith et al. "Effect of barren plateaus on gradient-free optimization". In: *Quantum* 5 (2021). ISSN: 2521327X. DOI: `10.22331/Q-2021-10-05-558`.

[7] Frank Arute et al. "Quantum supremacy using a programmable superconducting processor". In: *Nature* 574.7779 (2019). ISSN: 14764687. DOI: `10.1038/s41586-019-1666-5`.

[8] Chris Bernhardt. *Quantum Computing for Everyone*. The MIT Press, 2019. ISBN: 0262039257.

[9] Fernando G.S.L. Brandão, Aram W. Harrow, and Michał Horodecki. "Local Random Quantum Circuits are Approximate Polynomial-Designs". In: *Communications in Mathematical Physics* 346.2 (2016). ISSN: 14320916. DOI: `10.1007/s00220-016-2706-8`.

[10] C. Bravo-Prieto et al. *Variational quantum linear solver: A hybrid algorithm for linear systems*. 2019.

[11] Carlos Bravo-Prieto et al. "Quantum singular value decomposer". In: *Physical Review A* 101.6 (2020). ISSN: 24699934. DOI: `10.1103/PhysRevA.101.062310`.

[12] Keith A. Britt, Fahd A. Mohiyaddin, and Travis S. Humble. "Quantum accelerators for high-performance computing systems". In: *2017 IEEE International Conference on Rebooting Computing, ICRC 2017 - Proceedings*. Vol. 2017-January. 2017. DOI: `10.1109/ICRC.2017.8123664`.

[13] Harry Buhrman et al. "Quantum fingerprinting". In: *Physical review letters* 87.16 (2001). ISSN: 00319007. DOI: `10.1103/PhysRevLett.87.167902`.

[14] Enrico Cappanera. *Variational Quantum Linear Solver for Finite Element Problems: a Poisson equation test case*. TU Delft. 2021.

[15] M. Cerezo and Patrick J. Coles. "Higher order derivatives of quantum neural networks with barren plateaus". In: *Quantum Science and Technology* 6.3 (2021). ISSN: 20589565. DOI: `10.1088/2058-9565/abf51a`.

[16] M. Cerezo et al. "Cost function dependent barren plateaus in shallow parametrized quantum circuits". In: *Nature Communications* 12.1 (2021). ISSN: 20411723. DOI: `10.1038/s41467-021-21728-w`.

[17] M. Cerezo et al. *Cost-function-dependent barren plateaus in shallow quantum neural networks*. 2020.

[18] Andrew M. Childs. "Equation solving by simulation". In: *Nature Physics* 5.12 (2009). ISSN: 1745-2473. DOI: `10.1038/nphys1473`.

[19] Vasil S. Denchev et al. "What is the computational value of finite-range tunneling?" In: *Physical Review X* 6.3 (2016). ISSN: 21603308. DOI: `10.1103/PhysRevX.6.031015`.

[20] David P. DiVincenzo. *The physical implementation of quantum computation*. 2000. DOI: 10.1002/1521-3978(200009)48:9/11<771::AID-PROP771>3.0.CO;2-E.

[21] Yuxuan Du et al. "The Expressive Power of Parameterized Quantum Circuits". In: (Oct. 2018). DOI: 10.1103/PhysRevResearch.2.033125. URL: http://arxiv.org/abs/1810.11922%20http://dx.doi.org/10.1103/PhysRevResearch.2.033125.

[22] Chuck Easttom. *Quantum Computing Fundamentals*. Addison-Wesley, 2021.

[23] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. "A Quantum Approximate Optimization Algorithm". In: *Arxiv* (Nov. 2014). URL: http://arxiv.org/abs/1411.4028.

[24] Edward Farhi et al. *Quantum Computation by Adiabatic Evolution*. Tech. rep. Arxiv, 2000.

[25] Richard P. Feynman. "Simulating physics with computers". In: *International Journal of Theoretical Physics* 21.6-7 (1982). ISSN: 00207748. DOI: 10.1007/BF02650179.

[26] Daniel Gottesman. *The Heisenberg Representation of Quantum Computers*. Tech. rep. 1998.

[27] Harper R. Grimsley et al. "An adaptive variational algorithm for exact molecular simulations on a quantum computer". In: *Nature Communications* 10.1 (2019). ISSN: 20411723. DOI: 10.1038/s41467-019-10988-2.

[28] Stuart Hadfield et al. "From the quantum approximate optimization algorithm to a quantum alternating operator ansatz". In: *Algorithms* 12.2 (2019). ISSN: 19994893. DOI: 10.3390/a12020034.

[29] Aram Harrow and Saeed Mehraban. "Approximate unitary t-designs by short random quantum circuits using nearest-neighbor and long-range gates". In: (Sept. 2018). URL: http://arxiv.org/abs/1809.06957.

[30] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. "Quantum algorithm for linear systems of equations". In: *Physical Review Letters* 103.15 (2009). ISSN: 00319007. DOI: 10.1103/PhysRevLett.103.150502.

[31] Zoë Holmes et al. "Connecting Ansatz Expressibility to Gradient Magnitudes and Barren Plateaus". In: *PRX Quantum* 3.1 (2022). ISSN: 26913399. DOI: 10.1103/PRXQuantum.3.010313.

[32] Meng Jun Hu, Xiao Min Hu, and Yong Sheng Zhang. "Are observables necessarily Hermitian?" In: *Quantum Studies: Mathematics and Foundations* 4.3 (2017). ISSN: 21965617. DOI: 10.1007/s40509-016-0098-2.

[33] Hsin Yuan Huang, Kishor Bharti, and Patrick Rebentrost. "Near-term quantum algorithms for linear systems of equations with regression loss functions". In: *New Journal of Physics* 23.11 (2021). ISSN: 13672630. DOI: 10.1088/1367-2630/ac325f.

[34] D-Wave Systems Inc. *What is Quantum Annealing*. D-wave. 2021. URL: https://docs.dwavesys.com/docs/latest/c_gs_2.html.

[35] J. A. Jones and M. Mosca. "Implementation of a quantum algorithm on a nuclear magnetic resonance quantum computer". In: *Journal of Chemical Physics* 109.5 (1998). ISSN: 00219606. DOI: 10.1063/1.476739.

[36] Kyungtaek Jun et al. "Solving linear systems by quadratic unconstrained binary optimization on D-Wave quantum annealing device". In: SPIE-Intl Soc Optical Eng, Apr. 2021, p. 11. ISBN: 9781510642898. DOI: 10.1117/12.2591588.

[37] Abhinav Kandala et al. "Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets". In: *Nature* 549.7671 (2017). ISSN: 14764687. DOI: 10.1038/nature23879.

[38] Emanuel Knill, Gerardo Ortiz, and Rolando D. Somma. "Optimal quantum measurements of expectation values of observables". In: *Physical Review A - Atomic, Molecular, and Optical Physics* 75.1 (2007). ISSN: 10502947. DOI: 10.1103/PhysRevA.75.012328.

[39] Jonas M. Kübler et al. "An adaptive optimizer for measurement-frugal variational algorithms". In: *Quantum* 4 (2020). ISSN: 2521327X. DOI: 10.22331/q-2020-05-11-263.

[40] Oleksandr Kyriienko, Annie E. Paine, and Vincent E. Elfving. "Solving nonlinear differential equations with differentiable quantum circuits". In: *Physical Review A* 103.5 (2021). ISSN: 24699934. DOI: 10.1103/PhysRevA.103.052416.

[41] Ryan Larose. *Variational Quantum Linear Solver on Rigetti QCS*. 2020. URL: `https://github.com/rmlarose/rigettiVQLS/blob/master/vqls-notebook.ipynb`.

[42] Jun Li et al. “Hybrid Quantum-Classical Approach to Quantum Optimal Control”. In: *Physical Review Letters* 118.15 (2017). ISSN: 10797114. DOI: `10.1103/PhysRevLett.118.150503`.

[43] Lin Lin and Yu Tong. “Optimal polynomial based quantum eigenstate filtering with application to solving quantum linear systems”. In: *Quantum* 4 (2020). ISSN: 2521327X. DOI: `10.22331/Q-2020-11-11-361`.

[44] Hai Ling Liu et al. “Variational quantum algorithm for the Poisson equation”. In: *Physical Review A* 104.2 (2021). ISSN: 24699934. DOI: `10.1103/PhysRevA.104.022418`.

[45] Seth Lloyd. “Universal quantum simulators”. In: *Science* 273.5278 (1996). ISSN: 00368075. DOI: `10.1126/science.273.5278.1073`.

[46] Andrew Lucas. “Ising formulations of many NP problems”. In: *Frontiers in Physics* 2 (2014). ISSN: 2296424X. DOI: `10.3389/fphy.2014.00005`.

[47] Salvatore Mandrà, Helmut G. Katzgraber, and Creighton Thomas. “The pitfalls of planar spin-glass benchmarks: Raising the bar for quantum annealers (again)”. In: *Quantum Science and Technology* 2.3 (2017). ISSN: 20589565. DOI: `10.1088/2058-9565/aa7877`.

[48] Yuri Manin. *Computable and Noncomputable*. 1980.

[49] Andrea Mari, Thomas R. Bromley, and Nathan Killoran. “Estimating the gradient and higher-order derivatives on quantum hardware”. In: *Physical Review A* 103.1 (2021). ISSN: 24699934. DOI: `10.1103/PhysRevA.103.012405`.

[50] Jarrod R. McClean et al. “Barren plateaus in quantum neural network training landscapes”. In: *Nature Communications* 9.1 (2018). ISSN: 20411723. DOI: `10.1038/s41467-018-07090-4`.

[51] K. Mitarai et al. “Quantum circuit learning”. In: *Physical Review A* 98.3 (Sept. 2018). ISSN: 24699934. DOI: `10.1103/PhysRevA.98.032309`.

[52] Ashley Montanaro. *Quantum algorithms: An overview*. 2016. DOI: `10.1038/npjqi.2015.23`.

[53] Ashley Montanaro and Sam Pallister. “Quantum algorithms and the finite element method”. In: (Dec. 2015). DOI: `10.1103/PhysRevA.93.032324`. URL: `http://arxiv.org/abs/1512.05903%20http://dx.doi.org/10.1103/PhysRevA.93.032324`.

[54] Hector Jose Morrell and Hiu Yung Wong. “Study of using Quantum Computer to Solve Poisson Equation in Gate Insulators”. In: *International Conference on Simulation of Semiconductor Processes and Devices, SISPAD*. Vol. 2021-September. 2021. DOI: `10.1109/SISPAD54002.2021.9592604`.

[55] Kouhei Nakaji and Naoki Yamamoto. “Expressibility of the alternating layered ansatz for quantum computation”. In: *Quantum* 5 (2021). ISSN: 2521327X. DOI: `10.22331/q-2021-04-19-434`.

[56] Florian Neukart et al. “Traffic flow optimization using a quantum annealer”. In: *Frontiers in ICT* 4.DEC (2017). ISSN: 2297198X. DOI: `10.3389/fict.2017.00029`.

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

[58] Carlos Ortiz Marrero, Mária Kieferová, and Nathan Wiebe. “Entanglement-Induced Barren Plateaus”. In: *PRX Quantum* 2.4 (2021). ISSN: 26913399. DOI: `10.1103/PRXQuantum.2.040316`.

[59] Feng Pan, Keyang Chen, and Pan Zhang. “Solving the sampling problem of the Sycamore quantum supremacy circuits”. In: *Quant-ph* (Nov. 2021). URL: `http://arxiv.org/abs/2111.03011`.

[60] Hrushikesh Patil, Yulun Wang, and Predrag S. Krstić. “Variational quantum linear solver with a dynamic ansatz”. In: *Physical Review A* 105.1 (Jan. 2022). ISSN: 24699934. DOI: `10.1103/PhysRevA.105.012423`.

[61] Aidan Pellow-Jarman et al. “A comparison of various classical optimizers for a variational quantum linear solver”. In: *Quantum Information Processing* 20.6 (2021). ISSN: 15731332. DOI: `10.1007/s11128-021-03140-x`.

[62] PennyLane. *Understanding the Haar Measure*. 2021. URL: `https://pennylane.ai/qml/demos/tutorial_haar_measure.html`.

[63] PennyLane. *Unitary Designs*. 2021. URL: `https://pennylane.ai/qml/demos/tutorial_unitary_designs.html`.

[64] Arthur Pesah. *Quantum Algorithms for Solving Partial Differential Equations*. Tech. rep. University College London, 2020.

[65] John Preskill. “Quantum computing in the NISQ era and beyond”. In: *Quantum* 2 (2018). ISSN: 2521327X. DOI: `10.22331/q-2018-08-06-79`.

[66] Jonathan Romero et al. “Strategies for quantum computing molecular energies using the unitary coupled cluster ansatz”. In: *Quantum Science and Technology* 4.1 (2019). ISSN: 20589565. DOI: `10.1088/2058-9565/aad3e4`.

[67] Troels F. Rønnow et al. “Defining and detecting quantum speedup”. In: *Science* 345.6195 (2014). ISSN: 10959203. DOI: `10.1126/science.1252319`.

[68] Stefan H. Sack et al. “Avoiding barren plateaus using classical shadows”. In: *Arxiv* (Jan. 2022). URL: `http://arxiv.org/abs/2201.08194`.

[69] Yuki Sato et al. “Variational quantum algorithm based on the minimum potential energy for solving the Poisson equation”. In: *Physical Review A* 104.5 (2021). ISSN: 24699934. DOI: `10.1103/PhysRevA.104.052409`.

[70] Maria Schuld and Francesco Petruccione. *Machine Learning with Quantum Computers 2nd Edition*. Springer, 2021.

[71] Maria Schuld et al. “Evaluating analytic gradients on quantum hardware”. In: *Physical Review A* 99.3 (2019). ISSN: 24699934. DOI: `10.1103/PhysRevA.99.032331`.

[72] John Shalf. *The future of computing beyond Moore’s Law*. 2020. DOI: `10.1098/rsta.2019.0061`.

[73] Kunal Sharma et al. “Noise resilience of variational quantum compiling”. In: *New Journal of Physics* 22.4 (2020). ISSN: 13672630. DOI: `10.1088/1367-2630/ab784c`.

[74] Yangchao Shen et al. “Quantum implementation of the unitary coupled cluster for simulating molecular electronic structure”. In: *Physical Review A* 95.2 (2017). ISSN: 24699934. DOI: `10.1103/PhysRevA.95.020501`.

[75] Yaoyun Shi. “Both Toffoli and Controlled-NOT need little help to do universal quantum computing”. In: *Quantum Information and Computation* 3.1 (2003). ISSN: 15337146. DOI: `10.26421/qic3.1-7`.

[76] Seung Woo Shin et al. “How ”Quantum” is the D-Wave Machine?” In: (Jan. 2014). URL: `http://arxiv.org/abs/1401.7087`.

[77] Sukin Sim, Peter D. Johnson, and Alán Aspuru-Guzik. “Expressibility and Entangling Capability of Parameterized Quantum Circuits for Hybrid Quantum-Classical Algorithms”. In: *Advanced Quantum Technologies* 2.12 (2019). ISSN: 25119044. DOI: `10.1002/qute.201900070`.

[78] A. V. Uvarov and J. D. Biamonte. “On barren plateaus and cost function locality in variational quantum algorithms”. In: *Journal of Physics A: Mathematical and Theoretical* 54.24 (2021). ISSN: 17518121. DOI: `10.1088/1751-8121/abfac7`.

[79] Walter Vinci and Daniel A. Lidar. “Non-Stoquastic Interactions in Quantum Annealing via the Aharonov-Anandan Phase”. In: (Jan. 2017). DOI: `10.1038/s41534-017-0037-z`. URL: `http://arxiv.org/abs/1701.07494%20http://dx.doi.org/10.1038/s41534-017-0037-z`.

[80] Samson Wang et al. “Can Error Mitigation Improve Trainability of Noisy Variational Quantum Algorithms?” In: (Sept. 2021). URL: `http://arxiv.org/abs/2109.01051`.

[81] Samson Wang et al. “Noise-induced barren plateaus in variational quantum algorithms”. In: *Nature Communications* 12.1 (2021). ISSN: 20411723. DOI: `10.1038/s41467-021-27045-6`.

[82] Xin Wang, Zhixin Song, and Youle Wang. “Variational quantum singular value decomposition”. In: *Quantum* 5 (2021). ISSN: 2521327X. DOI: `10.22331/Q-2021-06-29-483`.

[83] Zhihui Wang et al. “Quantum approximate optimization algorithm for MaxCut: A fermionic view”. In: *Physical Review A* 97.2 (2018). ISSN: 24699934. DOI: `10.1103/PhysRevA.97.022304`.

[84] Wikipedia. *Bell state*. 2022. URL: `https://en.wikipedia.org/wiki/Bell_state`.

[85] Wikipedia. *Hadamard test (quantum computation)*. 2022. URL: `https://en.wikipedia.org/wiki/Hadamard_test_(quantum_computation)`.

[86] Wikipedia. *Swap test*. 2022. URL: `https://en.wikipedia.org/wiki/Swap_test`.

[87] Kevin Wills. *Quantum Computing for Structural Optimization*. TU Delft. 2020.

[88] Ronald de Wolf. *Quantum Computing: Lecture Notes*. University of Amsterdam. 2019. URL: `https://arxiv.org/abs/1907.09415`.

[89] Yiqing Zhou, E. Miles Stoudenmire, and Xavier Waintal. “What Limits the Simulation of Quantum Computers?” In: *Physical Review X* 10.4 (2020). ISSN: 21603308. DOI: `10.1103/PhysRevX.10.041038`.