UNIVERSITY
OF OSLO

Master’s thesis

The title of my thesis

Any short subtitle

Keran Chen

Physics
60 ECTS study points

Department of Physics
Faculty of Mathematics and Natural Sciences

Spring 2024

Keran Chen

The title of my thesis

Any short subtitle

Supervisor:
The Name

Abstract

Here come 3–6 sentences describing your thesis.

Contents

List of Figures

List of Tables

Preface

Here comes your preface, including acknowledgments and thanks.

Chapter 1

Introduction

As the name suggests, many-body physics is the study of systems with more than one particle, which most systems in nature are. As it turned out that the whole reductionism idea of "knowing a large system in term of its parts" alone is not enough for physicists to solve the many-body Schrödinger equation. Notoriously, for a system consisting of N particles, analytical solution to the Schrödinger equation does not exist for $N > 2$. Numerically finding an approximation for such systems is known as a np-complete problem, which scales exponentially in N [ref]. Almost a century ago, the Hartree-Fock theory was developed [ref]to reduce the complexity of these problems and it still remains a benchmark for other many-body methods today. Methods invented later are regarded as "Post-Hartree-Fock" methods, including the Coupled-Cluster method [ref]and Configuration-Interaction [ref]. Full Configuration-Interaction is regarded as the "exact" solution in the set of basis functions given, but it is computationally heavy [ref].

Physicist Richard Feynman had envisioned the possibility of simulating a quantum system with quantum computers which he proposed alongside [ref]. A few years later, David Deutsch showed that a "Universal Quantum Computer" can perfectly simulate every finitely realizable physical system [Deutsch85]. The Quantum Phase Estimation (QPE) algorithm was first proposed to estimate the eigenvalue of unitary operators [ref]. Though due to its lengthy circuit, and the limitation we have on Noisy intermediate-scale quantum (NISQ) era, running QPE circuit has not been possible on a quantum hardware [ref].

Fast forward forty years, today the field of quantum computation provides researchers a different approach to many-body problems and other fields of science [ref]. One comprised solution is the Variational Quantum Eigensolver (VQE) [peruzzo14]. This hybrid algorithm uses the power of both quantum and classical computers to ensure the quantum circuit to be executed has reasonable length. Variants of it have shown promising results in calculations of electronic structures in quantum chemistry [ref]. The parallels in electronic structure and nuclear structure have led to the application of the VQE algorithm in the latter.

The ansatz to a VQE is flexible, but it is the key to whether the correct ground state energy can be found and if it would be efficient. It is an active area of research to find the best ansatz for a given problem [ref]. In the past few years, VQE with a problem tailored ansatz by the name of ADAPT-VQE was invented with the goal of constructing ansatzs which contains information about the Hamiltonian of interest and avoid Barren plateaus [Grimsley19] [ref]. It has been shown that ADAPT-VQE produces better results than VQE algorithm with Unitary Couple-Clustered ansatz for molecular calculations [Grimsley19] and the nuclear shell model [Romero23], which inspired the application of ADAPT-VQE onto other models in nuclear physics.

In this thesis, we will first introduce the basic concepts of many-body physics, quantum computation and the VQE algorithm in chapter I. After that we introduce the implementation of the algorithms through Python library *Quanthon*. Then we will discuss the application of VQE in many-body physics in finding the ground state energy of a system, for models in nuclear physics and simple electronic structures. We will constrain us to running classical simulations of an ideal quantum computer for the most of the discussion, while keeping in mind that "In theory there is no difference between theory and practice, while in practice there is." more

Part I

Theory

Chapter 2

Quantum Mechanics

Notations

Dirac notation is used throughout the thesis, which follows Table 2.1. The hat on an operator $\hat{A}$ is often omitted when there is no ambiguity.

Table 2.1: Notations

Notation	Meaning
$\|\psi\rangle$	State vector, ket
$\langle\psi\|$	Dual vector of $\|\psi\rangle$, bra
$\langle\psi\|\phi\rangle$	Inner product of $\|\psi\rangle$ and $\|\phi\rangle$
$\|\psi\rangle\langle\phi\|$	Outer product of $\|\psi\rangle$ and $\|\phi\rangle$
$\hat{A}$	operator A
$\hat{I}$	Identity operator
$\langle\psi\|\hat{A}\|\psi\rangle$	Expectation value of $\hat{A}$ in $\|\psi\rangle$
$[\hat{A},\hat{B}]$	Commutator of $\hat{A}$ and $\hat{B}$
$\{\hat{A},\hat{B}\}$	Anticommutator of $\hat{A}$ and $\hat{B}$
$\hat{A}^\dagger$	Hermitian conjugate of $\hat{A}$
$\mathrm{Tr}\left(\hat{A}\right)$	Trace of $\hat{A}$
$A\otimes B$	Tensor product of $\hat{A}$ and $\hat{B}$
X,Y,Z	Pauli σ_x,σ_y and σ_z matrices.

2.1 Postulates

The postulates of quantum mechanics are the foundation of the theory, everything else follows as a consequence. They are the assumptions that the theory is built upon. The postulates are listed below:

Postulate 2.1.1

The state of a quantum system is described by a ket vector $|\psi\rangle$ in a Hilbert space $\mathcal{H}$.

Postulate 2.1.2

Every observable quantity is associated with a Hermitian operator $\hat{A}$ in $\mathcal{H}$. $\langle A\rangle = \langle\Psi|A|\Psi\rangle$ is the expectation value of $\hat{A}$ in the state $|\Psi\rangle$. The only possible result of a measurement of an observable $\hat{A}$ is one of the eigenvalues of $\hat{A}$.

Postulate 2.1.3

The time evolution of a wave function is governed by the Schrödinger equation:

$$i\hbar\frac{\partial}{\partial t}|\Psi(t)\rangle = \hat{H}|\Psi(t)\rangle \tag{2.1}$$

Postulate 2.1.4

The probability of measuring an observable $\hat{A}$ in the state $|\Psi\rangle$ is given by:

$$P(a) = \left|\langle\Psi|a\rangle^2\right| \tag{2.2}$$

where a is an eigenvalue of $\hat{A}$.

Postulate 2.1.5

The eigenvectors of a Hermitian operator form a complete basis, which means that any state vector can be expressed as a linear combination of the eigenvectors:

$$|\Psi\rangle = \sum_i c_i|\psi_i\rangle \tag{2.3}$$

where $\{\psi_i\}$ is a complete basis set.

2.2 The Schrödinger Equation

The Schrödinger equation is the fundamental equation of quantum mechanics. It describes the time evolution of a quantum system. The time evolution of a quantum system is unitary, which means that the norm of the state vector is conserved. The Schrödinger equation is given by:

$$i\hbar\frac{\partial}{\partial t}|\Psi(t)\rangle = \hat{H}|\Psi(t)\rangle \tag{2.4}$$

where $\hat{H}$ is the Hamiltonian operator of the system. The time independent Schrödinger equation is given by:

$$\hat{H}|\Psi\rangle = E|\Psi\rangle \tag{2.5}$$

where E is the energy of the system. The solution to the time independent Schrödinger equation is the eigenstates of the Hamiltonian operator.

2.3 Operators

According to Postulate 2.1.2, every observable can be represented by a Hermitian operator. In this section, we will discuss the properties of operators and their representations.

Theorem 2.3.1: Spectral Theorem

If $\hat{A}$ is a Hermitian operator in vector space V, then there exist an orthonormal basis of V consists of eigenvector of $\hat{A}$. Each eigenvalue of $\hat{A}$ is real. The spectral decomposition of a Hermitian operator $\hat{A}$ is given by:

$$\hat{A} = \sum_i a_i \left|\psi_i\right\rangle \left\langle\psi_i\right| \tag{2.6}$$

where a_i is the eigenvalue of $\hat{A}$ and $|\psi_i\rangle$ is the corresponding eigenvector.

2.3.1 Hermitian Operators

Definition 2.3.1: Hermitian Operators

An operator $\hat{A}$ is Hermitian if it satisfies:

$$\hat{A} = \hat{A}^\dagger \tag{2.7}$$

where $\hat{A}^\dagger$ is the Hermitian conjugate of $\hat{A}$. The Hermitian conjugate of an operator is defined as:

$$\langle\psi|\hat{A}^\dagger|\phi\rangle = \langle\phi|\hat{A}|\psi\rangle^* \tag{2.8}$$

The eigenvalues of a Hermitian operator are real, as

$$\begin{aligned}\langle\psi|a|\psi\rangle^* &= \langle\psi|\hat{A}^\dagger|\psi\rangle = \langle\psi|\hat{A}|\psi\rangle = \langle\psi|a|\psi\rangle \\ &\implies \langle\psi|a|\psi\rangle = a\,\langle\psi|\psi\rangle \in \mathbb{R} \\ &\implies a \in \mathbb{R} \qquad \text{since } \langle\psi|\psi\rangle \in \mathbb{R}\end{aligned} \tag{2.9}$$

where a is an eigenvalue of $\hat{A}$. This ensures that all measurable quantities and expectation values are real.

2.3.2 Unitary Operators

Definition 2.3.2: Unitary Operators

An operator $\hat{U}$ is unitary if it satisfies:

$$\hat{U}^\dagger\hat{U} = \hat{U}\hat{U}^\dagger = \hat{I} \tag{2.10}$$

This is an important property for quantum gates to have, as it ensures that the norm of the state vector is preserved.
Given an initial state $|\psi(0)\rangle$ and any operator $\hat{U}$. The evolution is given by

$$\hat{U}\,|\psi(0)\rangle = |\psi(t)\rangle$$

By requiring the norm of the state vector to be preserved, we have

$$\langle\psi(0)|\psi(0)\rangle = \langle\psi(t)|\psi(t)\rangle = \langle\psi(0)|U^{\dagger}U|\psi(0)\rangle$$

$$\implies U^{\dagger}U = \hat{I}$$

hence U is unitary.

2.3.3 Commutators and Anticommutators

Definition 2.3.3: Commutator

The commutator of two operators $\hat{A}$ and $\hat{B}$ is defined as:

$$[\hat{A},\hat{B}] = \hat{A}\hat{B} - \hat{B}\hat{A} \tag{2.11}$$

The anticommutator of two operators $\hat{A}$ and $\hat{B}$ is defined as:

$$\{\hat{A},\hat{B}\} = \hat{A}\hat{B} + \hat{B}\hat{A} \tag{2.12}$$

Operators in general do not commute. Two commuting operators have the same eigenvectors and can be measured simultaneously.
Important commutation relations are given in Appendix **??**.

2.4 Density Matrix

The density operator is useful to describe a mixed state, which contain classical uncertainties that cannot be described using superpositions.In finite dimensional space, the density operator is often called the *density matrix*.

Definition 2.4.1: Density Matrix

The density matrix ρ is defined as:

$$\rho = \sum_i p_i\,|\psi_i\rangle\,\langle\psi_i|\,, \tag{2.13}$$

where p_i is the probability of the state $|\psi_i\rangle$.

The expectation value of an observable $\hat{A}$ in a classical ensemble of states described by the density matrix ρ is given by:

$$\begin{aligned}
\langle A \rangle &= \sum_j P_j \langle \psi_j | A | \psi_j \rangle \\
&= \sum_j P_j \langle \psi_j | \sum_k |k\rangle \langle k | \hat{A} | \psi_j \rangle \\
&= \sum_{j,k} P_j \langle \psi_j | k \rangle \langle k | \hat{A} | \psi_j \rangle \\
&= \sum_{j,k} P_j \langle k | \hat{A} | \psi_j \rangle \langle \psi_j | k \rangle \\
&= \mathrm{Tr}\left(\hat{A}\rho\right).
\end{aligned} \tag{2.14}$$

For a pure state $|\psi\rangle$, the density matrix is the projector:

$$\rho = |\psi\rangle \langle \psi| . \tag{2.15}$$

2.5 Entanglement

For two states $|\psi\rangle$ and $|\phi\rangle$ in two Hilbert spaces $\mathbb{H}_1$ and $\mathbb{H}_2$ with basis $\{|u_j\rangle\}$ and $\{|w_j\rangle\}$ respectively, the tensor product $|\psi\rangle \otimes |\phi\rangle$ is a state in the Hilbert space $\mathbb{H}_1 \otimes \mathbb{H}_2$. This is called a *product state* and if

$$|\psi\rangle = \sum_j d_j |u_j\rangle ,$$

and

$$|\phi\rangle = \sum_k f_k |w_k\rangle ,$$

then

$$|\psi\rangle \otimes |\phi\rangle = \sum_{j,k} d_j f_k |u_j\rangle \otimes |w_k\rangle .$$

While an arbitrary state $|\Psi\rangle$ in the combined Hilbert space can be written as:

$$|\Psi\rangle = \sum_{j,k} c_{jk} |u_j\rangle \otimes |u_k\rangle ,$$

with coefficients c_{jk}. A product state has separable coefficients. If the coefficients c_{jk} are not separable, the state $|\Psi\rangle$ is called an entangled state.

2.6 Variational Principle

The variational method provides an upper bound on the ground state energy levels of a system. It is given by

$$E_0 \leq \langle \psi | \hat{H} | \psi \rangle . \tag{2.16}$$

It consists of two steps:

1. Choose an ansatz for the ground state.
2. Optimise the parameter of the ansatz to minimise the energy.

2.7 Many-Body Physics

In this section, we will discuss the many-body basis, indistinguishability and the second quantization formalism.

2.7.1 Many-Body Basis

If the single particle state is represented by $\{|\phi_i\rangle\}$, a single particle state $|\psi\rangle$ can be represented by

$$|\psi_{\text{1-particle}}\rangle = \sum_j c_j |\phi_j\rangle . \tag{2.17}$$

Then the N identical particle state can be written as a tensor product of the single particle basis:

$$|\psi_{\text{N-particle}}\rangle = \sum_{j_1,j_2,\ldots,j_N} c_{j_1,j_2,\ldots,j_N} |\phi_{j_1}\rangle \otimes |\phi_{j_2}\rangle \otimes \cdots \otimes |\phi_{j_N}\rangle . \tag{2.18}$$

2.7.2 Indistinguishability

Fundamental particles are indistinguishable. This means that the state of a system is invariant under the exchange of two particles. If $\hat{P}_{ij}$ is the particle exchange operator which exchanges state of particle i and particle j,

$$\hat{P}_{ij} |\psi(0,\ldots,i,\ldots,j,\ldots)\rangle = e^{i\phi} |\psi(0,\ldots,j,\ldots,i,\ldots)\rangle \tag{2.19}$$

Since the physics is invariant under the exchange of two particles, the state can only differ by a global phase. Applying the particle exchange operator again must give the exact state that we have started with.

$$\hat{P}_{ij}^2 |\psi(0,\ldots,i,\ldots,j,\ldots)\rangle = e^{2i\phi} |\psi(0,\ldots,i,\ldots,j,\ldots)\rangle = |\psi(0,\ldots,i,\ldots,j,\ldots)\rangle \tag{2.20}$$

This implies that $e^{2i\phi} = 1$, hence $e^{i\phi} = \pm 1$, which means that the state is either symmetric or antisymmetric under the exchange of two particles.

The principle of indistinguishability results in that the actual space for the both fermions and bosons are be smaller than the tensor product of the single particle Hilbert space.

2.7.3 Occupation Number (Second Quantisation) Notation

Because of the indistinguishability discussed in Section 2.7.2, the coefficients $\{c_{j_n}\}$ in Equation (2.18) are not independent and not all elements of the combined Hilbert space is a physical state. The occupation number notation provides a different way to represent the many-body state, also called second quantisation formalism. Since the fundamental particles are either symmetric or antisymmetric under exchange of particles, one could simply the number of particles in a given state, hence the occupation number notation. In the occupation number notation, Equation (2.18) becomes:

$$|\psi_{\text{N-particles}}\rangle = \sum_{n_1,n_2,\ldots,n_N} c_{n_1,n_2,\ldots,n_N} |n_1, n_2, \ldots, n_N\rangle . \tag{2.21}$$

Note that the coefficients $\{c_n\}$ are now independent and normalised.
While Equation (2.21) looks similar to Equation (2.18), they have very different physical interpretations. The state $|\phi_{j_n}\rangle$ is a state of the Nth particle and the state $|n_N\rangle$ gives the number of particles in the single particles state $|\phi_N\rangle$.

Fock Space

The occupation number notation introduces the possibility of having different number of the particles in a state, say

$$|\psi\rangle = |1,1,0\rangle + |1,0,0\rangle \, .$$

The space of all possible number of particles is called the Fock space. The Fock space the direct sum of the Hilbert space of all possible number of particles up to N . The Fock space is given by:

$$F_v(\mathbb{H}) = \bigoplus_{n=0}^{N} \hat{S}_v \mathbb{H}^{\otimes n}. \tag{2.22}$$

where $\hat{S}_v$ is the symmetrisation operator for bosons and the antisymmetrisation operator for fermions.

Operators in Second Quantisation

The fermionic creation is defined as:

Definition 2.7.1: Creation Operator

$$\hat{a}_i^\dagger |0\rangle \equiv |i\rangle \tag{2.23}$$

where $|0\rangle$ is the vacuum state and $|i\rangle$ is the state with one particle in the ith single particle state. The hermitian conjugate of the creation operator, $\hat{a}_i$ is the fermionic annihilation operator since

$$\hat{a}_i |i\rangle = |0\rangle \tag{2.24}$$

The number operator is given by:

$$\hat{n}_i = \hat{a}_i^\dagger \hat{a}_i \tag{2.25}$$

An Hamiltonian operator $\hat{H}$ with one-body and two-body interactions in the first quantisation formalism can be written in the second quantisation formalism as:

$$\begin{aligned} \hat{H} &= \sum_p \hat{T}_p + \sum_{p \neq q} \hat{V}_{pq} \\ &= \sum_p \sum_{jk} T_{jk} |u_j\rangle \langle u_k| + \sum_{p\neq q} \sum_{ijkl} V_{ijkl} |u_i, u_j\rangle \langle u_k, u_l| \\ &= \sum_{i,j} h_{ij} \hat{a}_i^\dagger \hat{a}_j + \sum_{i,j,k,l} v_{ijkl} \, \hat{a}_i^\dagger \hat{a}_j^\dagger \hat{a}_k \hat{a}_l \end{aligned} \tag{2.26}$$

2.7.4 Antisymmetric States

The Pauli exclusion principle states that no two fermions can occupy the same quantum state. This means that the wave function of a system of fermions must be antisymmetric under the exchange of two particles.
The wave function of a system of fermions is given by:

$$|\psi\rangle = \frac{1}{\sqrt{N!}} \sum_{\sigma \in S_N} (-1)^{\sigma} \hat{P}_{\sigma} |\psi(1), \psi(2), \ldots, \psi(N)\rangle \tag{2.27}$$

2.7.5 Slater Determinant

The Slater determinant is a way to antisymmetrise the many-body wave function. It is given by:

$$|\psi\rangle = \frac{1}{\sqrt{N!}} \begin{vmatrix} \psi_1(1) & \psi_2(1) & \ldots & \psi_N(1) \\ \psi_1(2) & \psi_2(2) & \ldots & \psi_N(2) \\ \vdots & \vdots & \ddots & \vdots \\ \psi_1(N) & \psi_2(N) & \ldots & \psi_N(N) \end{vmatrix} \tag{2.28}$$

where $\psi_i(j)$ is the ith single particle state with the jth particle.

Chapter 3

Quantum Computing

In this chapter we provide details of quantum computing and quantum algorithms. Starting by introducing the basic concepts of quantum computing, we then discuss the quantum algorithms used in this thesis. Unlike the previous chapter which provides details of the physics problem we are trying to solve, this chapter focuses on the algorithms and the application of it. Thus all physical objects such as the Hamiltonian or the Schrödingers equation will be treated as pure mathematical objects without physical implications.

3.1 Qubits

In analogy to the classical bit, the basic unit of information in classical computing, the basic unit of information in quantum computing is the quantum bit, or qubit. A qubit lives in a two dimensional Hilbert space which can be spanned by two basis vectors, theoretically of choice. Unlike the classical bit, the quantum nature of qubits allows them to be in a superposition of the two basis states. Conventionally, we denote the two basis state as $|0\rangle$ and $|1\rangle$, chosen to be the eigenstates of the Pauli-Z operator, such that:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}. \tag{3.1}$$

The state of a qubit can then be written as a linear combination of the two basis states,

$$|\psi\rangle = \alpha |0\rangle + \beta |1\rangle , \tag{3.2}$$

where α and β are complex numbers, with normalisation condition $\alpha^2 + \beta^2 = 1$.

3.1.1 Bloch Sphere

The state of a qubit can also be represented by a point on the Bloch sphere, as shown in Figure 3.1 . The Bloch sphere is a unit sphere, where the z direction

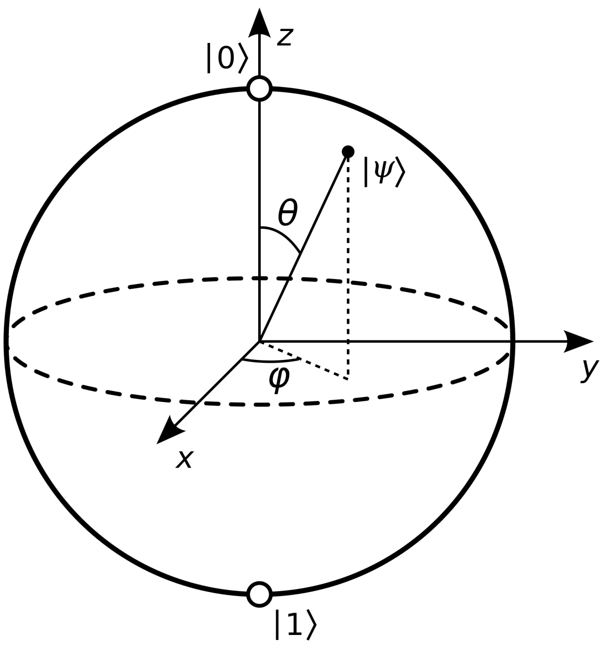


Figure 3.1: Bloch sphere representation of a qubit.

3.1.2 More than one qubit

Multi-qubit systems are no different from other many body quantum systems. The combined Hilbert space of n qubits is the tensor product of the individual Hilbert spaces of each qubit,

$$\mathbb{H} = \mathbb{H}_1 \otimes \mathbb{H}_2 \otimes \cdots \otimes \mathbb{H}_n. \tag{3.3}$$

The $|0 \ldots 0\rangle$ state is often referred to as the vacuum state of the system. It worth noting that qubits are assumed to be distinguishable unlike fermions or bosons. By convention, the qubit to the right is the first qubit.

3.2 Quantum Gates

Mathematically, quantum gates are a series of unitary operators in the operator space $\mathcal{H} \otimes \mathcal{H}^*$ which evolves the state. The unitary nature preserves the norm of the state vector, ensuring the probabilities sum to unity. Since not all gates correspond to an observable, they are not necessarily hermitian.

3.2.1 Single Qubit Gates

Pauli Gates

Pauli gates are Pauli matrices defined as Definition ??.

Definition 3.2.1: Pauli Matrices

he Pauli matrices are defined as

$$X \equiv \sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad Y \equiv \sigma_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad Z \equiv \sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{3.4}$$

The Pauli-X gate is also known as the NOT gate, which flips the state of the qubit.

$$X|0\rangle = |1\rangle, \tag{3.5}$$
$$X|1\rangle = |0\rangle. \tag{3.6}$$

The Pauli-Y gate flips the bit and multiply the phase by i.

$$Y|0\rangle = i|1\rangle, \tag{3.7}$$
$$Y|1\rangle = -i|0\rangle. \tag{3.8}$$

The Pauli-Z gate multiplies the phase by -1.

$$Z|0\rangle = |0\rangle, \tag{3.9}$$
$$Z|1\rangle = -|1\rangle. \tag{3.10}$$

Hadamard Gate

The Hadamard gate is defined as

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}. \tag{3.11}$$

It creates a superposition of the $|0\rangle$ and $|1\rangle$ states.

$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle), \tag{3.12}$$
$$H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle). \tag{3.13}$$

Phase Gates

The S-phase gate is usually denoted as S, and is defined as

$$S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}. \tag{3.14}$$

It multiplies the phase of the $|1\rangle$ state by i.

$$S|0\rangle = |0\rangle, \tag{3.15}$$
$$S|1\rangle = i|1\rangle. \tag{3.16}$$

Its inverse

$$S^\dagger = \begin{pmatrix} 1 & 0 \\ 0 & -i \end{pmatrix} \tag{3.17}$$

is known as the $S^\dagger$ gate which applies a $-i$ phase shift to the $|1\rangle$.

$$S^\dagger|0\rangle = |0\rangle, \tag{3.18}$$
$$S^\dagger|1\rangle = -i|1\rangle. \tag{3.19}$$

3.2.2 Two Qubit Gates

CNOT Gate

The CNOT gate is a two qubit gate which acts on two qubits, a control qubit and a target qubit. The CNOT gate is defined as

$$\text{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}. \tag{3.20}$$

It is often used to perform an linear entanglement on qubits.

$$\begin{aligned} \text{CNOT}\,|00\rangle &= |00\rangle\,, \\ \text{CNOT}\,|01\rangle &= |01\rangle\,, \\ \text{CNOT}\,|10\rangle &= |11\rangle\,, \\ \text{CNOT}\,|11\rangle &= |10\rangle\,. \end{aligned}$$

SWAP gate

The SWAP gate is a two qubit gate which swaps the state of two qubits. It is defined as

$$\text{SWAP} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{3.21}$$

$$\begin{aligned} \text{SWAP}\,|00\rangle &= |00\rangle\,, \\ \text{SWAP}\,|01\rangle &= |10\rangle\,, \\ \text{SWAP}\,|10\rangle &= |01\rangle\,, \\ \text{SWAP}\,|11\rangle &= |11\rangle\,. \end{aligned}$$

3.2.3 Pauli Strings

A Pauli string, such as $XIYZ$ is a tensor product of Pauli matrices acting on different qubits. The Pauli string $XIYZ$ is defined as

$$XIYZ \equiv X_0 \otimes I_1 \otimes Y_2 \otimes Z_3. \tag{3.22}$$

Hamiltonians are often rewritten or decomposed in terms of Pauli string as they can be easily implemented on quantum computers.

3.3 Quantum Circuits

A quantum circuit is a sequence of quantum gates applied to a set of qubits. For example, the bell state represented by $\frac{1}{\sqrt{2}}\left(|00\rangle + |11\rangle\right)$ is represented by Figure 3.2.

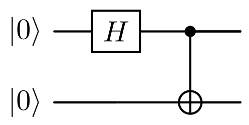
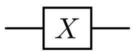
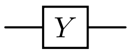
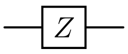
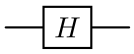
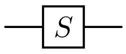
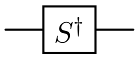
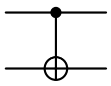
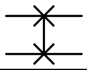


Figure 3.2: The quantum circuit which creates the bell state.

3.3.1 Quantum Circuit Diagrams

It is often convenient to represent quantum circuits using quantum circuit diagrams. Table 3.1 shows the common symbols used to present each component of a quantum circuit.

Table 3.1: Quantum Circuit Diagram

Gate	Symbol
X	
Y	
Z	
H	
S	
$S^\dagger$	
CNOT	
SWAP	

3.3.2 Measurement

A measurement is a discontinuous change of the state. Measurement necessarily collapses the state of the qubit to one of the basis states, with probability given by its coefficient. In general, if $\{|\psi_j\rangle\}$ is a complete set of basis, then a measurement on the state $|\psi\rangle = \sum_j c_j |\psi_j\rangle$ in this basis gives a result that can be described as

$$|\psi\rangle \to |\psi_j\rangle \quad \text{with probability} \quad |c_j|^2. \tag{3.23}$$

For a product state, the measurement result of any qubit would not affect the states of other qubits, i.e. the probability of each qubit is independent. On an entangled state, for example a bell state $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$, Measuring either qubit would collapse the other qubit into the same state.

The precision or relative error ϵ is defined as

Definition 3.3.1: Relative Error

If a is the true value of a quantity and a_{meas} is the measurement result, then the relative error ϵ is defined as

$$\epsilon = \left| \frac{a - a_{meas}}{a} \right| . \tag{3.24}$$

The number of times a measurement is repeated is called the number of shots. In general, a precision of ϵ requires $\mathcal{O}(1/\epsilon^2)$ shots as a result of statistics [**knill2007**].

3.4 Fermionic Encoding

The Pauli exclusion principle states that no two fermions can occupy the same quantum state, which means a state in a fermionic system can be represented by a binary string. One could map the fermionic creation and annihilation operators to gates as long as the mapping have the same commutation relation.

3.4.1 Jordan-Wigner Transformation

The Jordan-Wigner transformation is a mapping between fermionic operators and spin operators. If $\hat{a}_n^\dagger$ is the creation operator for the nth fermion,

$$\begin{aligned} \hat{a}_n^\dagger &\mapsto \frac{1}{2}\left[\prod_{j=0}^{n-1} -Z_j\right](X_n - iY_n) \\ \hat{a}_n &\mapsto \frac{1}{2}\left[\prod_{j=0}^{n-1} -Z_j\right](X_n + iY_n) \end{aligned} \tag{3.25}$$

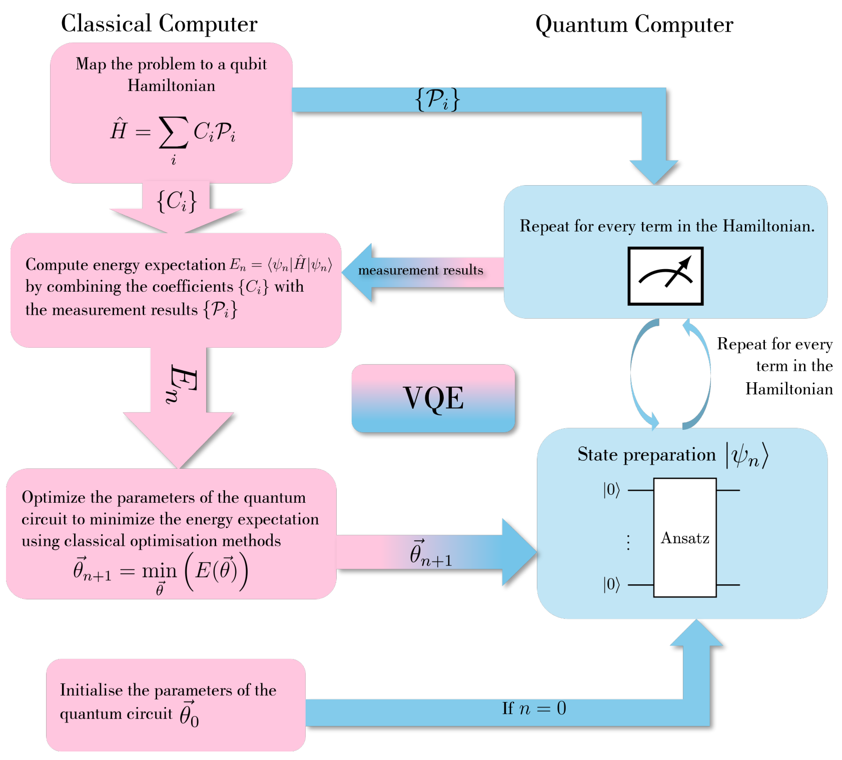
3.4.2 Pauli Decomposition

The Pauli matrices $\sigma_x(X), \sigma_y(Y), \sigma_z(Z)$, together with the identity operator I in $\mathbb{M}_{2\times 2}$ form a complete set of basis in $\mathbb{M}_{2\times 2}$. The tensor products of these operators form complete basis in high dimension of $\mathbb{M}_{2^n\times 2^n}$. If the Hamiltonian $\hat{H}$ of a system is given by a $2^n \times 2^n$ matrix, it could be mapped to qubit space using

$$\hat{H}_{\text{qubit}} = \sum_{\mathcal{P}} \frac{1}{2^n} Tr\left(\mathcal{P}\hat{H}\right), \tag{3.26}$$

where $\mathcal{P}$ is a Pauli string in $2^n \times 2^n$.

This uses only $\lceil \log_2 n \rceil$ qubits to map a $2^n \times 2^n$ matrix.



Classical Computer
Quantum Computer
Map the problem to a qubit Hamiltonian
$\hat{H} = \sum_i C_i \mathcal{P}_i$
$\{\mathcal{P}_i\}$
$\{C_i\}$
Repeat for every term in the Hamiltonian.
Compute energy expectation $E_n = \langle\psi_n|\hat{H}|\psi_n\rangle$ by combining the coefficients $\{C_i\}$ with the measurement results $\{\mathcal{P}_i\}$
measurement results
E_n
VQE
Repeat for every term in the Hamiltonian
State preparation $|\psi_n\rangle$
$|0\rangle$
Ansatz
$|0\rangle$
Optimize the parameters of the quantum circuit to minimize the energy expectation using classical optimisation methods
$\vec{\theta}_{n+1} = \min_{\vec{\theta}} \left(E(\vec{\theta})\right)$
$\vec{\theta}_{n+1}$
Initialise the parameters of the quantum circuit $\vec{\theta}_0$
If $n = 0$

Figure 3.3: Illustration of the VQE algorithm.

3.5 Quantum Algorithms

3.5.1 Variational Quantum Eigensolver

The variational quantum eigensolver is a hybrid quantum algorithm that can be realised on a NISQ quantum computer. It consist of an ansatz, aka. a parameterised circuit. The ansatz prepares a state which is then measured to estimate the energy expectation value. The acts as the objective function for the classical optimiser which updates the parameter iteratively to find the optimal parameter which minimises the energy utilising the variational principle [**peruzzo2014**]. The VQE algorithm is illustrated in Figure 3.3. The hybrid and variational nature of the algorithm allows to be run on NISQ devices and the classical optimisation reduces the length and depth of the quantum circuit.

Second Quantised Hamiltonian

The Hamiltonians we try to solve with VQE are usually with Hamiltonians in the form of Equation (3.27).

$$\hat{H} = \sum_{pq} h_{pq} \hat{a}_p^\dagger \hat{a}_q + \frac{1}{2} \sum_{pqrs} h_{pqrs} \hat{a}_p^\dagger \hat{a}_q^\dagger \hat{a}_r \hat{a}_s. \tag{3.27}$$

where h_{pq} and h_{pqrs} are the one and two body integrals, given by

$$\langle p|\hat{h}_0|q\rangle$$

and

$$\langle pq|\hat{V}|rs\rangle$$

where $\hat{h}_0$ is the one-body operator and $\hat{V}$ is the two-body (interaction) operator.

3.5.2 Ansatz

The form of the ansatz for the VQE is crucial for the convergence of the VQE. A good ansatz should be able to represent the ground state of the system, so that convergence to the ground state energy is possible; span only the necessary section of the Hilbert space, thus reducing the optimisation complexity; as well as being shallow, since we are still in the NISQ era.

Hardware Efficient Ansatz

Hardware efficient ansatz is a class of ansatzes that uses single qubit rotation gates and linear entanglement CNOT gates that are available to quantum computers [ref]. Figure 3.4 illustrates an example circuit for a hardware efficient ansatz with four qubits. The single qubit rotation gates are parameterised and these parameters are optimised classically to search for the minimum. Note that there could be many other different kinds of hardware efficient ansatz, such as the RyRz ansatz, which uses Ry and Rz gates, or the Ry ansatz, which uses only Ry gates.

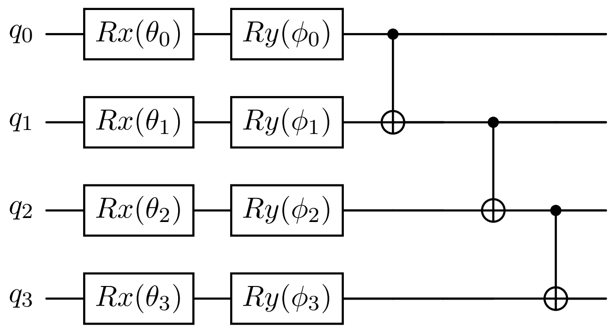


Figure 3.4: Example of the hardware efficient ansatz in a our-qubit circuit.

The hardware efficient ansatz usually have relatively short circuits compared to other ansatz and are easy to implement on quantum computers since they consist only of gates that are directly available on the hardware. One downside of the hardware efficient ansatz is that there is no information about the Hamiltonian we are trying to solve, and since the ansatz does not exhaust the entire Hilbert space nor should it, it may not be able to represent the ground state of the system. However, it is almost always a good starting point due to the simplicity of the ansatz and serves as a good benchmark for more complex ansatz. Although it seem like an ab initio approach, the hardware efficient ansatz can actually successfully represent the ground state of many system, as we will show in Chapter 6. The reason why this worked at all is due to the fact that the hardware efficient ansatz consists of rotation gates `Rx, Ry, Rz` and CNOT gates, and these gates form a universal gate set. According to the Solovay-Kitaev theorem, any unitary operator can be approximated to arbitrary precision using a finite number of gates from a universal gate set [**Kitaev1997**]. This means that the gates involved in the hardware efficient ansatz can theoretically represent any unitary operator, and thus any state in the Hilbert space.

3.5.3 Adaptive, Problem Tailored VQE

Grimsley et a. proposed the Adaptive, Problem Tailored VQE (ADAPT-VQE) algorithm [**grimsley2019**] which aims to provide an adaptive ansatz structure which can capture information about the Hamiltonian. The ADAPT-VQE uses an evolving ansatz which is updated every iteration by appending a new operator onto the circuit chosen from an predefined operator pool without draining the pool. The ADAPT-VQE follows the follow steps:

1. Define an operator pool $\{A\}$.
2. Calculate the gradient of energy with respect to each operator in the pool.
3. Select the operator with the largest gradient, A , append $e^{\theta A}$ onto the circuit.
4. Optimise all the parameters in the circuit using VQE.
5. Repeat steps 2 to 4 until the gradient of all operators is below a threshold.

After n ADAPT iterations, the ansats should look like:

$$|\psi_n\rangle = e^{\theta_0 A_0} e^{\theta_1 A_1} \dots e^{\theta_n A_n} |\psi_0\rangle .$$

Selection Criteria

Step 2 and 3 of the Adapt VQE requires the calculation of the gradient of the energy with respect to the new parameter. The gradient of the operator $\left(\frac{\partial E}{\partial \theta}\right)_{\theta=0}$ is given by:

$$\left(\frac{\partial E}{\partial \theta}\right)_{\theta=0} = \left(\frac{\partial}{\partial \theta}\right) \langle\psi_n| e^{-\theta A} H e^{\theta A} |\psi_n\rangle_{\theta=0} \tag{3.28}$$

$$= \langle\psi_n| \left(-A e^{-\theta A} H e^{\theta A}\right) + \left(e^{-\theta A} H A e^{\theta A}\right) |\psi_n\rangle_{\theta=0} \tag{3.29}$$

$$= \langle\psi_n| - AH + HA |\psi_n\rangle \tag{3.30}$$

$$= \langle\psi_n| [H, A] |\psi_n\rangle . \tag{3.31}$$

The operator $e^{\theta A_i}$ will be appended if

$$\left(\frac{\partial E}{\partial \theta_i}\right)_{\theta_i=0} > \left(\frac{\partial E}{\partial \theta_j}\right)_{\theta_j=0}, \quad \forall A_j \in \{A\}. \tag{3.32}$$

The ADAPT-VQE algorithm is illustrated in Figure 3.5 .

3.5.4 qubit-ADAPT-VQE

The qubit-ADAPT-VQE is a hardware efficient variant of the ADAPT-VQE where the operator pool is chosen to contain only odd Pauli strings, i.e. Pauli strings with odd number of Y operators. Since in the case where the Hamiltonian has time reversal symmetry, the Hamiltonian is real, choosing imaginary Pauli strings causes the analytical expression of the gradient given in Equation (3.28) to vanish. By choosing only odd operators the ansatz also stays real throughout the ADAPT iterations. A complete pool is defined as a pool that can transform the reference

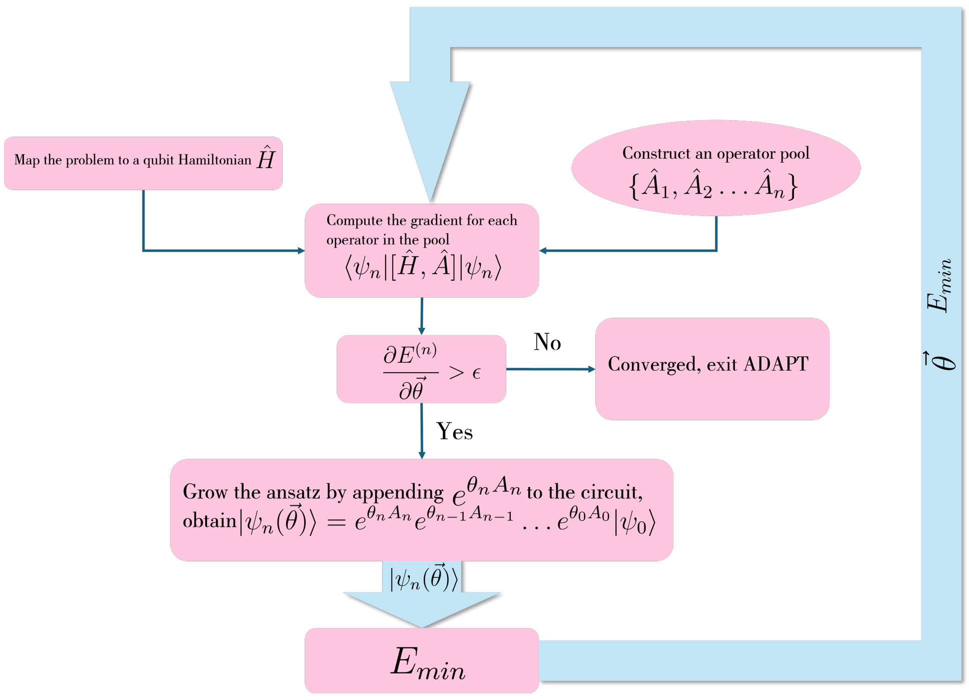

Map the problem to a qubit Hamiltonian $\hat{H}$
Construct an operator pool $\{\hat{A}_1, \hat{A}_2 \dots \hat{A}_n\}$
Compute the gradient for each operator in the pool $\langle\psi_n|[\hat{H}, \hat{A}]|\psi_n\rangle$
$\frac{\partial E^{(n)}}{\partial \vec{\theta}} > \epsilon$
No
Converged, exit ADAPT
Yes
Grow the ansatz by appending $e^{\theta_n A_n}$ to the circuit, obtain $|\psi_n(\vec{\theta})\rangle = e^{\theta_n A_n} e^{\theta_{n-1} A_{n-1}} \dots e^{\theta_0 A_0} |\psi_0\rangle$
$|\psi_n(\vec{\theta})\rangle$
E_{min}
$\vec{\theta}$ E_{min}

Figure 3.5: Illustration of the ADAPT-VQE algorithm.

state to any real state. To create any arbitrary real state in the n qubit Hilbert space requires only $2^n - 1$ real parameters. If $\{A_i\}$ is such that for any arbitrary state $|\psi\rangle$, $A_i|\psi\rangle$ form a complete basis, then $\{A_i\}$ is referred to as a complete basis of operators. The minimum size of a complete pool for n qubit has size $2n - 1$, which scales only linearly in number of qubits [**tang2021**].

We will present the two familier of minimal complete pool proven by [**tang2021**] and refer them as the the V pool and G pool from now on. A family of complete operator pool, $\{V_j\}_n$, where V_j is an operator in the pool and n is the number of qubits, can be constructed recursively as follows,

$$\{V_j\}_2 = \{iY_1Z_2, iI_1Y_2\},$$

$$\{V_j\}_n = \{\{V_k\}_{n-1}Z_n,\ iI_1I_2\dots Y_n,\ iI_1I_2\dots Y_{n-1}I_n\}.$$

The $\{V_j\}_n$ can be mapped onto a different family of operator pool, $\{G_j\}_n$ which contains iY on every single qubit and two qubit operator iY_kZ_{k+1} on adjacent qubits k and $k+1$,

$$\{G_j\}_n = \{iY_1I_2\dots I_n,\ iI_1Y_2\dots I_n,\ \dots,\ \ iI_1\dots I_{n-1}Y_n,$$

$$iY_1Z_2I_3\dots I_n,\ iI_1Y_2Z_3I_4\dots I_n,\ \dots, iI_1\dots I_{n-2}Y_{n-1}Z_n\}.$$

3.5.5 Barren Plateaus

The VQE algorithm amongst many other hybrid quantum algorithms which relies on doing optimisation classically and often uses what is called a *Random Parameterised Quantum Circuit(RPQC)* with the form

$$U(\vec{\theta}) = U(\theta_1 \dots \theta_L) \prod_{l=1}^{L} U_l(\theta_l) W_l, \tag{3.33}$$

where $U_l(\theta_l)$ is the parameterised unitary operator of the circuit and W_l is the unitary operator that does not depend on any parameter. Take the hardware efficient ansatz circuit from Figure 3.4 as an example: the U_l would be the single qubit rotation gates and the W_l would be the CNOT gates, as shown in Figure 3.6.

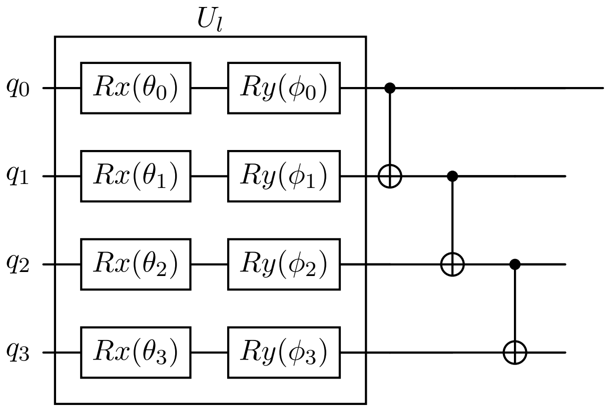


Figure 3.6: The U_l and W_l in a RPQC.

This is a common form for the ansatz because it allows easy calculation of the gradients

$$\frac{\partial E\left(\vec{\theta}\right)}{\partial \theta_l} = \frac{\partial \langle \psi(\vec{\theta})|H|\psi(\vec{\theta})\rangle}{\partial \theta_l} = i \left\langle 0 \middle| U_-^\dagger \left[V_k, U_+^\dagger H U_+\right] U_- \middle| 0 \right\rangle \tag{3.34}$$

where $U_- \equiv \prod_{l=0}^{k-1} U_l\left(\theta_l\right) W_l$ and $U_+ \equiv \prod_{l=k+1}^{L} U_l\left(\theta_l\right) W_l$ [**mcclean2018**]. Mcclean et al. also showed that for a large number of random circuits, the average value of the gradient and the variance of the gradient decrease exponentially with the number of qubits [**mcclean2018**]. Unlike classical deep neural networks, which scales as $\mathcal{O}(log(1/\epsilon))$, the cost of estimating the gradient scales with $\mathcal{O}(\frac{1}{\epsilon^\alpha})$ where ϵ is the deviation from the average value desired and α is arbitrarily small. The optimisation process is akin to a random walk if ϵ is the size of the gradient and the number of measurements is not on the order of $1/\epsilon^\alpha$. This phenomenon is known as the barren plateau problem, as the gradient has exponentially small probability of random walking out of the plateau [**mcclean2018**].

Part II

Implementation

Chapter 4

Quanthon: Quantum Computing in Python

This chapter details the design and implementation of a quantum computing library in Python [ref], namely `Quanthon`. The library is can be found at https://pypi.org/project/Quanthon/ and the source code at https://github.com/moyasui/Quanthon.

While existing quantum computing libraries such as Qiskit, Circ and OpenFermion [ref]are usually better integrated with hardware, more optimised and contain more functionalities, writing an entire library from scratch can be beneficial pedagogically by forbidding the existence of black boxes as well as allowing more freedom in structures and conventions. One example of such is the ordering of qubits. Let A, B, C, D be single qubit operations, most literature uses $ABCD$ to mean $A_1B_2C_3D_4$ i.e. applying the gate A on the first qubit, B on the second etc. This, however, is inconsistent with the computer science convention of representing numbers using bit strings, where the most significant bit is to the left. For example, the decimal 4 is usually written as 100 in binary representation. Most quantum computing libraries adheres to the binary representation convention by having the most significant bit to the left, therefore logically the aforementioned 4-qubit gate should have been written as $DCBA$, which sadly is not. There is no physical reason to this choice and it is purely conventional, hence no reason to compromise for it. In Quanthon, the most significant bit is to the right, and the order of operators/gates are written normally, in the same order.

Quanthon is also designed to be minimalistic and with the focus of physicists in mind. The library is designed to be used for quantum simulations and solving physical problems, following the most fundamental principal rather than specialise in, e.g. electronic systems. The library is also designed to be easily extensible, with the ability to add custom gates and operators, algorithms and models.

Quanthon will continued to be maintained and expanded after the conclusion of this master's project.

4.1 Architecture

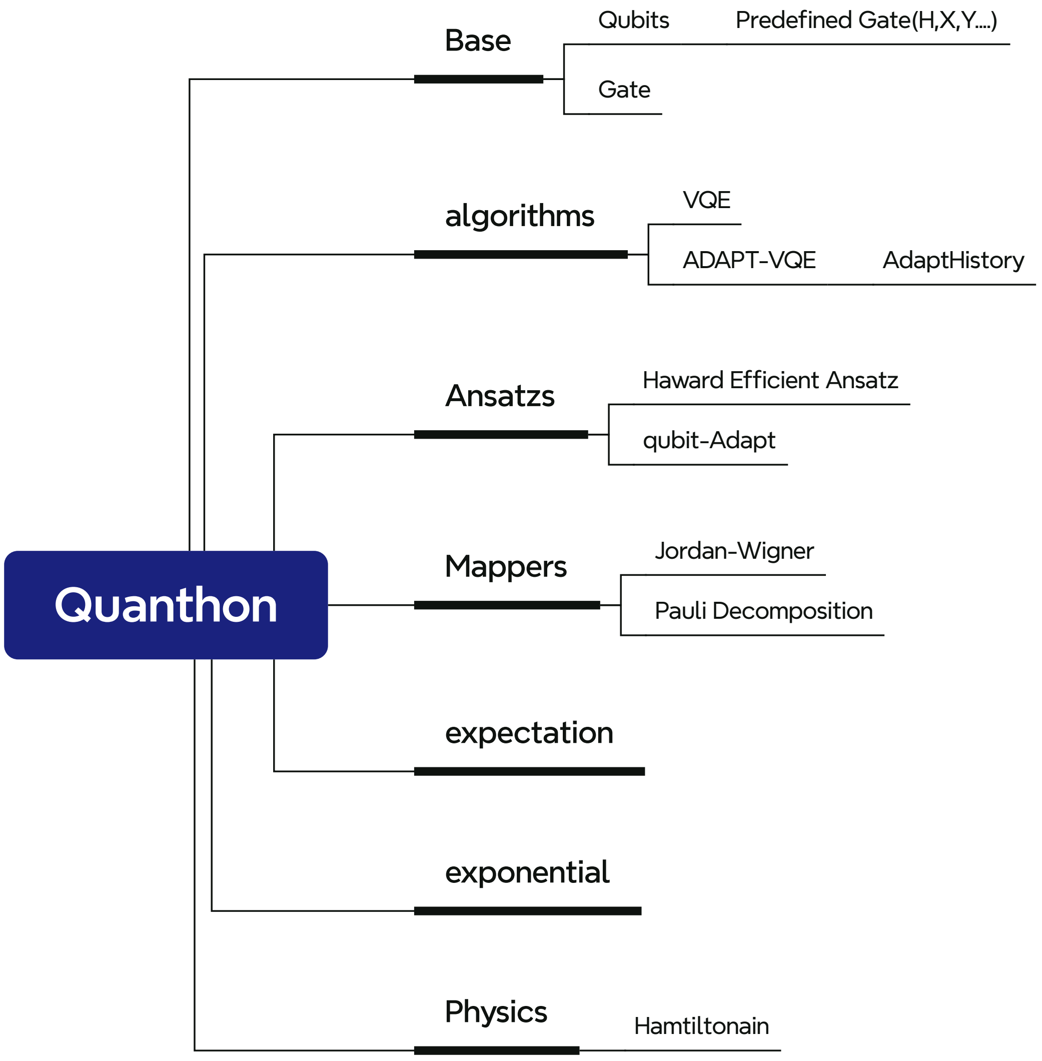


Figure 4.1: Architecture of Quanthon.

Figure 4.1 shows the architecture of Quanthon. Quanthon is built with doing quantum simulations and solving physical problems in mind, and it is divided into a few modules and several standalone functions. The basic module is the `base` module, this is where all the circuitry and gates are defined. The `algorithms` and `Ansatz` are heavily focused on the Variational Quantum Eigensolver. The `expectation` module contains functions that calculates the expectation value of an operator and `exponential` creates a circuit which exponentiates a Pauli string. The `Physics` module is used for storing the integrals of a second quantised Hamiltonian. The `Mappers` contains functions for transforming the Hamiltonian from different forms in terms of Pauli operators. The `utils` and `mappers_utils` modules contains helper functions.

4.2 Basic Elements of Quantum Computing

In theory, all quantum computation can be simulated using only these basic elements we introduces in this section, qubits and gates as described in Chapter 3. This section explains the structure for `base.py`, including the `Gate` class and `Qubits` class.

4.2.1 Qubits

A `Qubits` object is the most fundamental unit for quantum computation. It contains information about

- the number of qubits, `Qubits.n_qubit`,
- the current circuit, `Qubits.circuit`,
- the current state of the qubits, `Qubits.state`,
- the history of all the gates applied to the circuit, which is not cleared when the circuit is reset `Qubits.gate_history`, .

Initialise a `Qubits` object of 2 qubits to the $|0\rangle$ state by calling

```
        from Quanthon import Qubits
        q = Qubits(2)
```

To manipulate the circuit by appending gates to the circuit,

```
        q.H(0) # appends the Hadamard gate (a \texttt{Gate} object) on
           the $ 0th $ qubit to q.circuit. \
        q.CNOT(0,1) # appends the CNOT with control qubit $ 0 $  and
           target qubit $ 1 $ to q.circuit. \
        q.run() # apply all the gates in q.circuit
        q.draw() # generate code for drawing the circuit in Quantikz \
           missingref.
```

Below is all currently available gates as methods of the `Qubits` class

- `H` (Hadamard gate),
- `X,Y,Z` (Pauli matrices),
- `Rx, Ry, Rz` (single qubit rotation gates),
- `Sdag`, $S^\dagger$, the Hermitian conjugate of the S-phase gate,
- `CNOT` (controlled-not) gate,
- `SWAP` (the swap gate).

To simulate a classical measurement with n shots, where the probability is given by amplitude squared using the Born rule .

```
        counts = q.measure(n_shots=10000)
```

where `count` contains the simulated result for 10000 of the same circuit `q`. The output is $2^n \times 2$ array, the first column is the count corresponding to the state in binary in the second column.

```
>>> counts
array([[5025, '00'],
       [0, '01'],
       [0, '10'],
       [4975, '11']], dtype=object)
```

Other available methods that could be useful include

- `reset` to reset the state of the qubits to $|0\rangle$,
- `set_state` which sets the state of the qubits to a given state, also checks if the state is valid,
- `reset_circuit` which resets the circuit to an empty list,
- `copy` which returns a copy of the `Qubits` object with the same states but not gates in the circuit, this is useful when repeated circuit preparation is needed for measurement,
- `_to_comp_basis` which returns the state in the more readable bra-ket notation when the print statement is called. For example, the following code gives

  ```
          >>> q = Qubits(2)
          >>> print(q)
  ```

  $$\text{Qubit(s) in state:}1.00 + 0.00j\,|00\rangle$$

- `count_gates` which returns the length of the circuit
- `count_cnots` which returns the number CNOTgates currently in the circuit. `draw` provides visualisation of the circuit, returns a Quantikz [ref]string that represents the circuit.

4.2.2 Shot Noise

In this section we will numerically show the error of state tomography for different number of shots and compare their precisions. We first prepare a state $|\psi\rangle$ such that

$$|\psi\rangle = \frac{1}{2}\left(|00\rangle + |01\rangle + |10\rangle + |11\rangle\right)$$

We now measure the state $|\psi\rangle$ for $[4^2, 4^3, 4^4, 4^5, 4^6, 4^7, 4^8, 4^9]$ shots and compare the error, as well as to the theoretical result from Subsection 3.3.2. If the precision required is ϵ then the number of shots required $\propto \frac{1}{\epsilon^2}$. Then every time the number of shots is increased by a factor of 4, the error should half in size.

Figure 4.2 and 4.3 demonstrates relationship between the error and the number of shots agree quite well with the theoretical result: $N \propto \mathcal{O}(\frac{1}{\epsilon^2})$. In fact, from the gradient of Figure **??** we can estimate errors from measurements with $\epsilon = \sqrt{\frac{1}{N}}$. Later we will show that the error from shot noise is the limiting factor for the accuracy of algorithms.

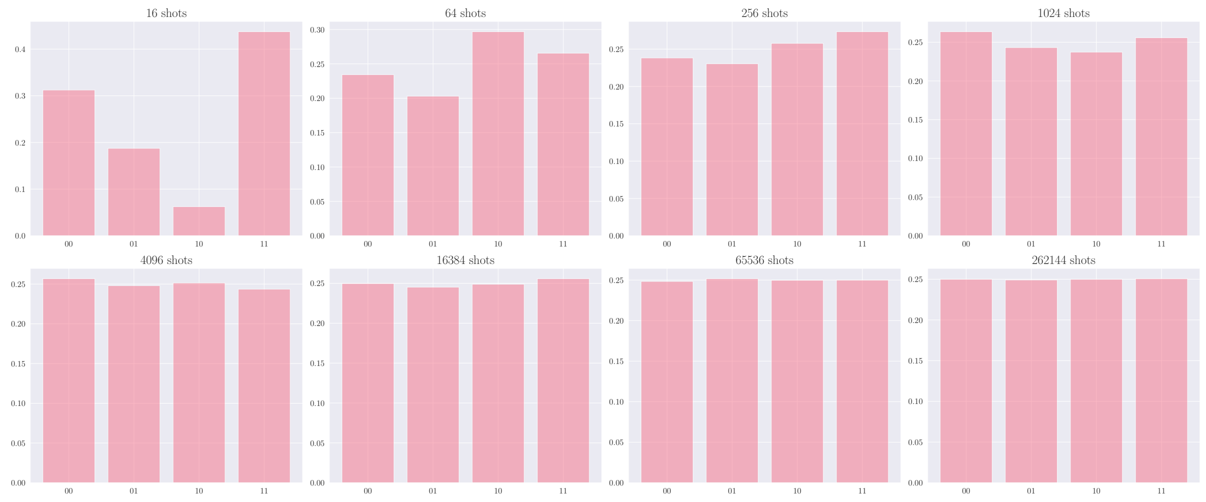


Figure 4.2: Histograms of measurement results for state $|\psi\rangle$ with different number of shots. The horizontal axis labels the states being measured and the vertical axis shows probability for measuring the corresponding state.

4.2.3 Gate

The `Qubits` class contains common predefined gates as in Subsection 4.2.1. To construct a custom gate, which can be parameterised, specify name, the number of qubits being acted on and the matrix representation of the gate or a function which takes in parameters and generates the gate matrix. The matrix must be unitary according to Definition 2.3.2.
Below is an example to how to construct a `Rz` gate for a two qubit circuit.

```python
import numpy as np
from Quanthon.base import Gate
from Quanthon.utils import make_op_mat

def Rz(theta, n, i):
        """
        args:
            theta: float, the angle of rotation in radians,
            n: int, the number of qubits in the system,
            i: int, the index to apply the gate to.

        returns:
            mat: the matrix representing the rotation of qubit i by
                theta about the z-axis.
        """

    rz = np.cos(theta/2)*np.eye(2) - 1j*np.sin(theta/2)*np.array
        ([[0,1],[1,0]])
    mat = make_op_mat(rz, i) # this expands the matrix to the size of
        the full system
    return mat

n = 2
i = 0
# non-parameterised
theta = 0.5 * np.pi
mat = Rz(theta, n, i)
rz_gate = Gate("Rz", mat, n)
```

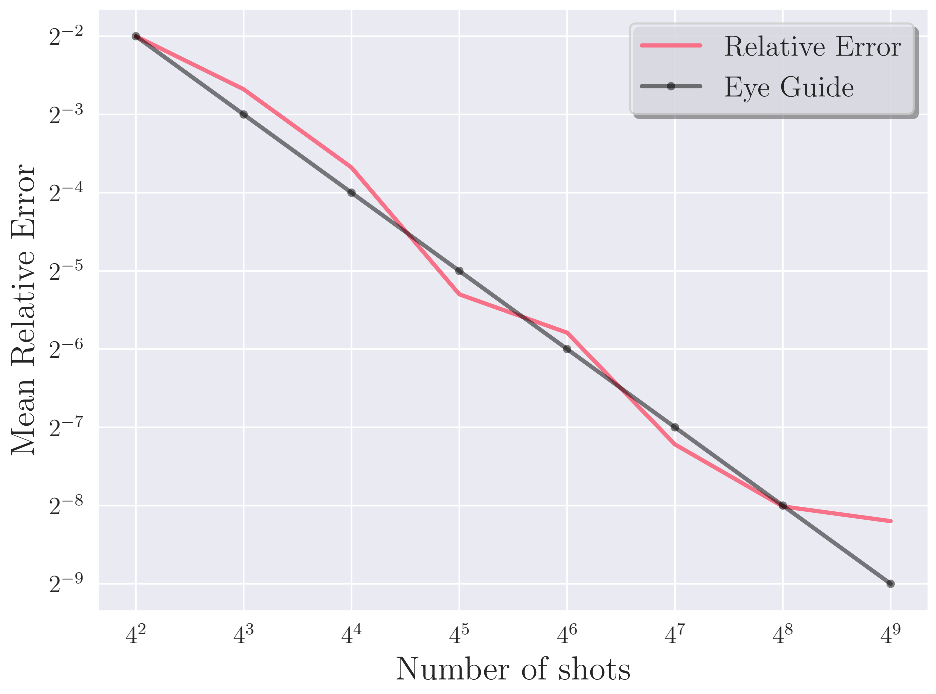


Figure 4.3: Mean relative error for each shot. The red line is the result of numerical simulation using the `Quanthon` library and the grey line is an eye guide with $y = 1/\sqrt{x}$.

```

# Parameterised
rz_gate_param = Gate("Rz", lambda theta: Rz(theta,i,n), n,
    is_parametrised=True)
```

To use the custom gate,

```
        q = qubits(2)
        # apply the non-parameterised Rz gate
        q.circuit.append(rz_gate) # append the gate here without any
            parameter
        q.run()

        # or the parameterised gate
        q.state = rz_gate_param.act(q.state, 0.5 * np.pi)
```

Note that custom gate does not currently support drawing.

4.3 Expectation

The expectation value of an operator A is usually given as

$$\langle\psi|A|\psi\rangle \, .$$

In a real quantum computer this value must be estimated through repeated measurements on the state of the qubits in different basis. Since the Pauli matrices together with the identity operator I form a basis for $\mathcal{M}_{2\times 2}$, any operator in this space can be written in terms of a linear combination of these operators using methods given in Appendix A.2.

Let P be a Pauli string, then the expectation value of P is given by

$$\langle\psi|P|\psi\rangle = \langle\psi|U^\dagger ZU|\psi\rangle \tag{4.1}$$

One must find an unitary operator U such that $U^\dagger ZU$ is equal to the basis we are trying to measure in the one-qubit case, and $ZI\ldots I$ in the multiple-qubit case. When the unitary transformation U is applied to the circuit of which we want measure the expectation value, the eigenvalues are preserved and the expectation value can be calculated in the new basis.

4.3.1 Single Qubit

In general, the one qubit Hamiltonian can be written as

$$H = aI + bX + cY + dZ, \quad a, b, c, d \in \mathbb{C}. \tag{4.2}$$

Given the following relations relating the Pauli X and Y operator with the Pauli Z.

$$X = HZH \tag{4.3}$$
$$Y = -S^\dagger HZHS \tag{4.4}$$

Then the expectation value is

$$\begin{aligned}\langle\psi|H|\psi\rangle &= a\langle\psi|I|\psi\rangle + b\langle\psi|X|\psi\rangle + c\langle\psi|Y|\psi\rangle + d\langle\psi|Z|\psi\rangle && (4.5)\\ &= a\underbrace{\langle\psi|\psi\rangle}_{1} + b\langle\psi|HZH|\psi\rangle + c\langle\psi|HS^\dagger ZSH|\psi\rangle + d\langle\psi|Z|\psi\rangle\,. && (4.6)\end{aligned}$$

The `expectation` function estimates the expectation values of a qubit Hamiltonian.

In the case of a single qubit, to rotate the computation basis to another Pauli basis, simply apply the H gate for when there is an X in the Hamiltonian, and $S^\dagger$ gate followed by H gate when there is a Y.

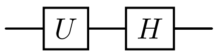


Figure 4.4: slice doesn't work The single qubit circuit which rotates a state to be measured in the X basis.

4.3.2 Multiple Qubits

The computation basis is the Pauli $Z\underbrace{I\ldots I}_{n-1}$ basis for a n qubit quantum circuit. When there are more than one qubit in the system, if the operator contains only one non-identity operator on the first qubit, then similar strategy applies — apply the `H` or the `Sdag` followed by `H` gate on the first qubit.

In the case when there are more than one non-identity, the `CNOT` gate must be applied to such qubits to disentangle them. Consider the smallest non-trivial case, the Pauli string ZZ, the unitary transform which rotates the ZZ basis to the ZI basis is the $CNOT_{10}$ gate,

$$CNOT(1,0) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} = CNOT(1,0)^\dagger,$$

and

$$CNOT(1,0)(ZZ)CNOT(1,0) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{4.7}$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} = ZI. \tag{4.8}$$

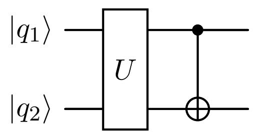


Figure 4.5: The circuit to rotate the qubits into the ZZ basis.

In cases where the non-identity gates are not in the first positions, a series of SWAP gates are used to swap the state of the qubits with neighbouring qubits.

$$SWAP(0,1)(IZ)SWAP(0,1) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.9}$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} = ZI. \tag{4.10}$$

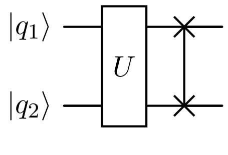


Figure 4.6: The circuit to rotate the qubits into the IZ basis.

After the rotation, multiple measurements are taken to estimate the state of the qubits, since the eigenvalues are preserved under unitary transformations, the eigenvalues are the same as that of the $ZI\dots I$ state. For an n qubit system, the states $|0\rangle$ to $|2^{n-1}\rangle$ have eigenvalues 1 and the rest have eigenvalues -1.

The `expectation` function handles all the procedures above for all combinations of Pauli strings. The code snippet outlines the usage of the function.

```python
from Quanthon import expectation
q = Qubits(2)
H = [("ZZ", 0.5), ("XI", 0.5)] # example Hamiltonian, must be
    given in [(operator, coefficient), ...] format.
energy = cal_expectation(q, hamiltonian, n_shots=10000)
```

4.4 Physics

4.4.1 Hamiltonian

The `Hamiltonian` class stores the overlap integrals for one and two-body. These integrals need to have been calculated. In the case of molecular system, these integrals can usually be obtained through quantum chemistry packages such as `PySCF` [ref].

4.5 Mappers

4.5.1 Jordan-Wigner

The Jordan-Wigner transformation maps the fermionic operators to qubit operators. The transformation is given in Section 3.4.1.

The second quantised Hamiltonians are usually in the form of Equation (3.27). We keep the integrals general without considering the symmetry of the system, i.e., $p, q, r, s \in [0, N-1]$ for N basis functions for both the one- and two-body integrals. The mapping costs N qubits. The treatment depend on the value and order of the indices and weather it is a one- or two-body term.

4.5.2 One-Body

The transformation of the one-body term $a_p^\dagger a_q$using Equation (3.25) is

$$a_p^\dagger a_q = \left(\frac{1}{2}\left[\bigotimes_{n=0}^{p-1} Z_n\right](X_p - iY_p)\right)\left(\frac{1}{2}\left[\bigotimes_{n=0}^{q-1} Z_n\right](X_q + iY_q)\right), \tag{4.11}$$

Four relations are useful in the derivation of the later expressions:

$$(X - iY)Z = -iY + X = (X - iY) \tag{4.12}$$
$$Z(X - iY) = iY - X = -(X - iY) \tag{4.13}$$
$$(X + iY)Z = -iY - X = -(X + iY) \tag{4.14}$$
$$Z(X + iY) = iY + X = (X + iY) \tag{4.15}$$

For $p = q$

When $p = q$, using properties of Pauli matrices in Appendix A, the Jordan-Wigner transform of the term Equation (4.11) becomes:

$$\begin{aligned}
a_p^\dagger a_q &= a_p^\dagger a_p \\
&= \left(\frac{1}{2}\left[\bigotimes_{n=0}^{p-1} Z_n\right](X_p - iY_p)\right)\left(\frac{1}{2}\left[\bigotimes_{n=0}^{p-1} Z_n\right](X_p + iY_p)\right) \\
&= \frac{1}{4}\left[\bigotimes_{n=0}^{p-1} I_n\right](X_p - iY_p)(X_p + iY_p) \\
&= \frac{1}{4}\left[\bigotimes_{n=0}^{p-1} I_n\right]\left(X_p^2 - iY_iX_i + iX_iY_i + Y_p^2\right) \\
&= \frac{1}{4}\left[\bigotimes_{n=0}^{p-1} I_n\right](2I_p + 2iZ_p) \\
&= \frac{1}{2}\left[\bigotimes_{n=0}^{p-1} I_n\right](I_p - Z_p)
\end{aligned} \tag{4.16}$$

which translates to two Pauli strings. In the case of N qubits $\underbrace{I\dots I}_{N}$ and $\underbrace{I\dots I}_{p}Z_p\underbrace{l\dots I}_{N-p-1}$ respectively.

For $p < q$

In the case where $p < q$, the Z operators on qubits on p and between p and q do not cancel out as in the $p = q$ case, Equation (4.11) becomes:

$$\begin{aligned}
&= \frac{1}{4}\left[\bigotimes_{n=0}^{p-1} I_n\right](X_p - iY_p)\,Z_p\left[\bigotimes_{n=p+1}^{q-1} Z_n\right](X_q + iY_q) \\
&= \frac{1}{4}\left[\bigotimes_{n=0}^{p-1} I_n\right](X_p - iY_p)\left[\bigotimes_{n=p+1}^{q-1} Z_n\right](X_q + iY_q) \\
&= \frac{1}{4}\left[\bigotimes_{n=0}^{p-1} I_n\right]\left[\bigotimes_{n=p+1}^{q-1} Z_n\right](X_p - iY_p)(X_q + iY_q) \\
&= \frac{1}{4}\left[\bigotimes_{n=0}^{p-1} I_n\right]\left[\bigotimes_{n=p+1}^{q-1} Z_n\right](X_pX_q - iY_pX_q + iX_pY_q + Y_pY_q)\,.
\end{aligned} \tag{4.17}$$

This translates to four Pauli strings:

$$\underbrace{I\dots I}_{p} X_{p} \underbrace{Z\dots Z}_{q-p-1} X_q \underbrace{I\dots I}_{N-q} \tag{4.18}$$

$$-i\underbrace{I\dots I}_{p} Y_{p} \underbrace{Z\dots Z}_{q-p-1} X_q \underbrace{I\dots I}_{N-q} \tag{4.19}$$

$$i\underbrace{I\dots I}_{p} X_{p} \underbrace{Z\dots Z}_{q-p-1} Y_q \underbrace{I\dots I}_{N-q} \tag{4.20}$$

$$\underbrace{I\dots I}_{p} Y_{p} \underbrace{Z\dots Z}_{q-p-1} Y_q \underbrace{I\dots I}_{N-q} \tag{4.21}$$

For $p > q$

The equation is the same but with p and q swapped. The Pauli strings are also the same but since now $p > q$, the operator with index q is on the left hand side.

$$\underbrace{I\dots I}_{p} X_{q} \underbrace{Z\dots Z}_{p-q-1} X_p \underbrace{I\dots I}_{N-q} \tag{4.22}$$

$$-i\underbrace{I\dots I}_{p} X_{q} \underbrace{Z\dots Z}_{p-q-1} Y_p \underbrace{I\dots I}_{N-q} \tag{4.23}$$

$$i\underbrace{I\dots I}_{p} Y_{q} \underbrace{Z\dots Z}_{p-q-1} X_p \underbrace{I\dots I}_{N-q} \tag{4.24}$$

$$\underbrace{I\dots I}_{p} Y_{q} \underbrace{Z\dots Z}_{p-q-1} Y_p \underbrace{I\dots I}_{N-q} \tag{4.25}$$

4.5.3 Two-Body

The two-body term $a_p^\dagger a_q^\dagger a_r a_s$ is more complicated. The general Jordan-Wigner transform is.

$$\begin{aligned} a_p^\dagger a_q^\dagger a_r a_s = &\left(\frac{1}{2}\left[\bigotimes_{n=0}^{p-1} Z_n\right](X_p - iY_p)\right)\left(\frac{1}{2}\left[\bigotimes_{n=0}^{q-1} Z_n\right](X_q - iY_q)\right) \\ &\left(\frac{1}{2}\left[\bigotimes_{n=0}^{r-1} Z_n\right](X_r + iY_r)\right)\left(\frac{1}{2}\left[\bigotimes_{n=0}^{s-1} Z_n\right](X_s + iY_s)\right) \end{aligned} \tag{4.26}$$

If the outer indices are not greater or less than the inner indices, the term will different by a factor of -1 due to the creation operators being on the right hand side, i.e. $p > q \wedge r > s$ or $p < q \wedge r < s$.
The results of the transformation depends on the values of the indices p, q, r, s.

For $p = q$ or $r = s$

According to definition of the creation and annihilation operators,

$$a_p^\dagger a_p^\dagger = 0,$$

and

$$a_q a_q = 0.$$

Therefore if the creation or the annihilation operators have the same indices, this term does not contribute.

For $p = r \neq q = s$ or $p = s \neq q = r$

Since $p = r$ and $q = s$, all the Z operators cancel out since there are always an even number of Z operators on every qubit, then Equation (4.26) simplifies to:

$$\begin{aligned} a_p^\dagger a_q^\dagger a_r a_s &= a_p^\dagger a_q^\dagger a_p a_q \\ &= \frac{1}{4}\left[\bigotimes_{n=0}^{p-1} I_n\right](I_p - Z_p)\left[\bigotimes_{n=p+1}^{q-1} I_n\right](I_q - Z_q) \\ &= \frac{1}{4}\left[\bigotimes_{n=0}^{p-1} I_n\right]\left[\bigotimes_{n=p+1}^{q-1} I_n\right](I_pI_q - Z_pI_q - I_pZ_q + Z_pZ_q) \end{aligned} \tag{4.27}$$

For $p \neq q \neq r \neq s$

In the case where all the indices are unique, we first sort the indices so that $a < b < c < d \equiv$ `sorted(p,q,r,s)`.

$$\begin{aligned} a_p^\dagger a_q^\dagger a_r a_s = \frac{1}{16}\left[\bigotimes_{n=0}^{a-1} I_n\right](X_a \pm iY_a)\, Z_a\left[\bigotimes_{n=a+1}^{b-1} Z_n\right](X_b \pm iY_b) \\ \left[\bigotimes_{n=b+1}^{c-1} I_n\right](X_c \pm iY_c)\, Z_c\left[\bigotimes_{n=c+1}^{d-1} Z_n\right](X_d \pm iY_d)\,, \end{aligned} \tag{4.28}$$

where the $\pm$ depends on whether the associated operator is a creation or annihilation operator. These would result in 16 unique Pauli strings of products of the $(X \pm iY)$ terms.

For 3 unique indices

The expression for the case where there are 3 unique indices changes depending on which index is repeated. Since the two creation and annihilation operators do not have the same index as discussed in Subsection 4.5.1, the three qubits that are being acted on must contain a creation, an annihilation and a product of both, which is given by $(X - iY)(X + iY) = (I - Z)$ following similar derivation of Equation 4.16.

Example Code

The Jordan-Wigner transformation can be done by calling the `jordan_wigner` function, which takes in an instance of `Hamiltonian` and handles all above cases then returns a list of Pauli strings with coefficients.

```python
from Quanthon import Hamiltonian, jordan_wigner

# Generate random one- and two-body integrals
one_body = np.random.rand(n,n) + 1j*np.random.rand(n,n)
one_body = one_body + one_body.conj().T

two_body = np.random.rand(n,n,n,n) + 1j*np.random.rand(n,n,n,n)
two_body = two_body + two_body.conj().T

hamiltonian = Hamiltonian(one_body, two_body)

qubit_hamiltonian = jordan_wigner(hamiltonian)
```

4.5.4 Pauli Decomposition

In case where a Hamiltonian is already written as matrix, Pauli decomposition as in Equation (3.26) can be used for a mapping which scales with $\lceil \log_2(N) \rceil$ for a $N \times N$ matrix. The `pauli_decomposition` function takes in a matrix and returns a list of Pauli strings with coefficients. In the case where Hamiltonian is not of size $2^n \times 2^n$, 0's are padded to all non-diagonal entries and 999 to the diagonal entries to make it of that shape. 999 is an arbitrary large number so that the additional rows and columns do not affect the ground state energy.

```python
from Quanthon import pauli_decomposition
matrix = np.random.rand(4,4) + 1j*np.random.rand(4,4)
matrix = matrix + matrix.conj().T
pauli_strings = pauli_decomposition(matrix)
```

4.6 Exponentials

Taking the exponential of a Pauli string is an inevitable and common step in quantum computing. Being able to take the exponential of a Pauli string allows us to implement, for example, the ADAPT-VQE. We have implemented two different algorithms for creating circuit for the exponentials that are both exact and equivalent mathematically, the first is the so-called Staircase algorithm and the second is the Inverted Staircase algorithms [ref].

4.6.1 Staircase Algorithm

The Staircase algorithm is an algorithm that decomposes the exponential into a rotation about the Z axis and CNOT gates. We will show that this is equivalent to the exponential itself. Given an exponential of a Pauli string $e^{-i\theta P}$, where θ is a real parameter. The exponential of a Pauli operator σ_j for the one qubit case can be written as

$$
\begin{aligned}
e^{-i\theta\sigma_j} &= \cos(\theta\sigma_j) + i\sin(\theta\sigma_j) && (4.29)\\
&= \sum_k \left((-1)^k \frac{(\theta\sigma_j)^{2k}}{(2k)!} \right) + i \sum_k \left((-1)^k \frac{(\theta\sigma_j)^{2k+1}}{(2k+1)!} \right) && (4.30)\\
&= \cos(\theta)I - i\sin(\theta)\sigma_j, && (4.31)
\end{aligned}
$$

since

$$\sigma_j^{2k} = I \text{ and } \sigma_j^{2k+1} = \sigma_j \quad \text{for } k \in \mathbb{N} \tag{4.32}$$

This, incidentally, is the rotation gate $R_{\sigma_j}(2\theta)$. This is true for Pauli strings acting on any number of qubits, since the matrix product act on only one of the subspace and operators acting on different qubits commute, hence Equation (4.32) holds. In general, for any Pauli string P

$$e^{-i\theta P} = \cos(\theta)I - i\sin(\theta)P. \tag{4.33}$$

Hence, the exponential of a Pauli string is a rotation about the axis specified by the Pauli string in the N dimensional hypersphere, where N is the number of qubits, analogous to the Bloch sphere for a single qubit. However, performing a multi-qubit rotation is not as straightforward as the single qubit case. Using similar methods to rotation of basis, we could find U such that

$$U(e^{-i\theta P})U^\dagger = e^{-i\theta Z^{\otimes N}}.$$

The results are similar to Equation (4.3) — for every X in the Pauli string,

$$U = U^\dagger = H.$$

Here we chose for Y a slightly different unitary operator

$$U = Y_L \equiv R_z\left(\frac{-\pi}{2}\right)H,$$

and

$$U^\dagger = Y_R \equiv= HRz(\frac{-\pi}{2}).$$

Finally, to covert a rotation about the $Z^{\otimes k}$ to single qubit rotations R_z we again choose $U_2 = \{\text{CNOT}(j, j+1)\}$ for $j \in \{0, \ldots, N-2\}$ so that

$$U\left(e^{-i\theta Z^{\otimes N}}\right)U^\dagger = e^{-i\theta I \ldots IZ}.$$

In the case where there are non-connected operators in the Pauli string, the SWAP gate is used instead of the CNOT gate. For example, if the Pauli string $P = XYZI$, the unitary transformation U would be

$$H\text{CNOT}(0,1) \otimes Y_L\text{CNOT}(1,2) \otimes \text{SWAP}(2,3) \otimes I.$$

In quantum circuit notation, this is

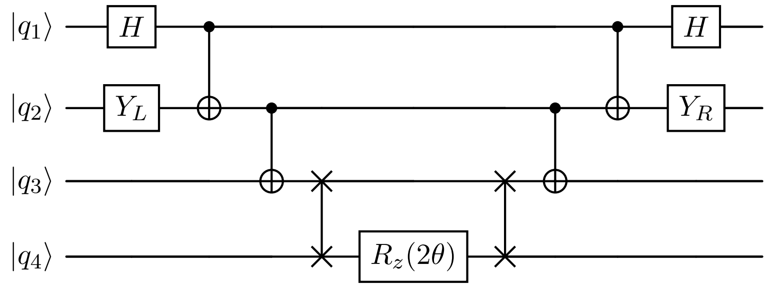


Figure 4.7: The circuit to exponentiate $-i\theta XYZI$ with the Staircase algorithm.

The structure of the circuit looks like a staircase made of CNOT gates.

4.6.2 Inverted Staircase Algorithm

The inverted staircase algorithm was proposed to reduce the number of single qubit gates, using R_x as the single qubit rotation instead of the R_z gate, as well the equivalent circuit shown in Figure **??**, In the Inverted Staircase algorithm, we try to find a mapping U such that

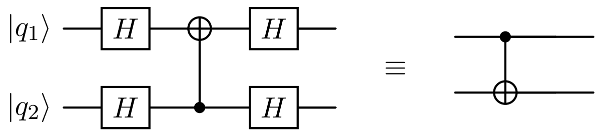


Figure 4.8: Equivalent circuit used in the Inverted Staircase Staircase.

$$Ue^{-i\theta P}U^{\dagger} = e^{-i\theta I \ldots X}.$$

Then for every Z in the Pauli string, apply a H gate on both the left and the right side of the CNOT (or SWAP) staircase, and for every Y apply a $R_z(\frac{\pi}{2})$ gate since

$$R_z\left(\frac{\pi}{2}\right)XR_z\left(-\frac{\pi}{2}\right) = Y.$$

Using the same example Pauli string $XYIZ$ as in the Staircase algorithm example, the circuit with Inverted Staircase algorithm is shown in Figure 4.9.

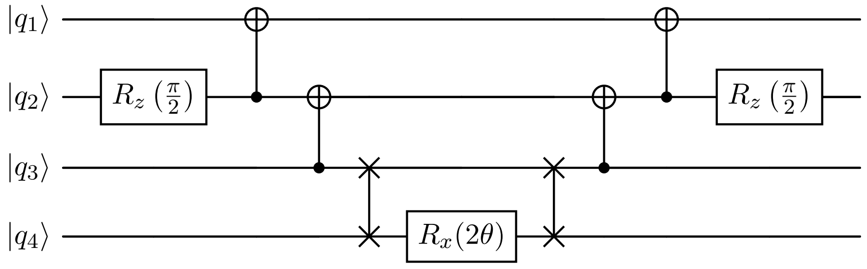


Figure 4.9: The circuit to exponentiate $-i\theta XYIZ$ using the Inverted Staircase algorithm.

Comparing Figure 4.7 and Figure 4.9, the Inverted Staircase algorithm has fewer single qubit gates (3) than the Staircase algorithm (7). Being able to reduce the number of single qubit gates is important in the NISQ era. Table 4.1 shows the comparison of the two algorithms for different Pauli strings. It is not difficult to see from Table 4.1 that if we ignore the one qubit rotation in the center for both algorithms, the Inverted Staircase algorithm is more efficient in terms of the number of single qubit gates by having the opposite number of gates for X and Z operators compared to the Staircase algorithm, but half as many for Y operators.

4.6.3 Example Code

The function `exponential_pauli` takes in a `Qubits` object, the Pauli string to exponentiate and the parameter θ for the keyword `coefficient`. It appends the gates using either the staircase or the inverted staircase algorithms to the circuit. Users can specify the algorithm by passing in either "staircase" or "inverted_staircase" to the keyword `method`. The code snippet below shows how to use the function. Note that this function currently only support two or more qubits, in the one qubit case, a normal rotation gate can be used.

Pauli String	Length	Number of SQG (Staircase)	Number of SQG (Inverted Staircase)
XX	2	7	3
YY	2	11	7
ZZ	2	3	7
$XYXY$	4	19	11
$XYIZ$	4	13	11
$IXIZ$	4	9	9
$XYZXYZ$	6	23	19
Total		85	67

Table 4.1: Comparison of the number of single qubit gates for the Staircase and Inverted Staircase algorithms.

```python
from Quanthon import exponential_pauli
q = Qubits(4)
# the resulting circuit will be appending onto the circuit of
   the qubits object
exponential_pauli(q, "XYIZ", np.pi/2, method="
   inverted_staircase")
```

4.7 Ansatz

The Ansatz module contains different ansatz which are used together with VQEs and potentially other variational algorithms. Ansatz is the based class for all fixed-form ansatz, such as the HardwareEfficientAnsatz, and QubitAdaptAnsatz is an ansatz which evolves every iteration.

4.7.1 Hardware Efficient Ansatz

The Hardware Efficient Ansatz is popular since it is easy to implement on a quantum computer. Consists of single qubit rotation gates and linear entanglement gates with CNOT, construct a circuit similar shown in Figure 3.4. The HardwareEfficientAnsatz class contains methods create_circuit and run which are called when passed as an ansatz to the VQE class.

```python
from Quanthon import HardwareEfficientAnsatz
ansatz = HardwareEfficientAnsatz(n_qubits=2, rep=2)
```

4.7.2 RyAnsatz

The RyAnsatz is also a type of hardware efficient ansatz except with only R_y gates instead of both R_x and R_y . The RyAnsatz class contains then same methods as the HardwareEfficientAnsatz class, and is used in the same way. Note that the number of parameters is n_{qubits} per rep instead of $2n_{\text{qubits}}$ as in the HardwareEfficientAnsatz class. The RyAnsatz typically achieves the same accuracy as the HardwareEfficientAnsatz but with only half as many parameters, and it is therefore more recommended for NISQ devices.

4.7.3 QubitAdaptAnsatz

QubitAdaptAnsatz is an ansatz which evolves with every iteration and it is used together with the `AdaptVQE` in Subsection 4.8.2.

The `QubitAdaptAnsatz` class takes in the number of qubits and a pool, which could either be "V", "Vx" or "G", or a custom pool given as a list of Pauli strings, default to "V". The V and G pool are given in Section 3.5.4, where "Vx" is has the same structure as "V" but all the Y gates are replaced by X gates.

The `append_op` method is called in the `AdaptVQE` class to append the parameterised version of the exponential of the selected operator. The `QubitAdaptAnsatz` class contains also a `Qubits` object at which the circuit is appended to.

There are also two different methods for running the circuit, `run` and `run_without_update`, the former updates the state and the latter returns the results while retaining the original state. These options are important for the AdaptVQE algorithm.

An initial state can also be passed in during initialisation, if not, the `Qubits` object will be initialised to the $\frac{1}{\sqrt{2^{N-1}}}(|0\dots0\rangle + \dots + |1\dots1\rangle$ state. The code snippet below shows how to use the `QubitAdaptAnsatz` class.

```
from Quanthon import QubitAdaptAnsatz
ansatz = QubitAdaptAnsatz(n_qubits=2, pool="G")
```

4.8 Algorithms

This section explains the structure for `algorithms.py`, including `VQE` and `ADAPT-VQE`.

4.8.1 VQE

The `VQE` class is used an implementation of the Variational Quantum Eigensolver which can be used with any fixed-form ansatz, in this example we will use the `qubit_hamiltonian` and `HardwareEfficientAnsatz` created in code blocks in Subsection 4.7.1 and Sub-subsection 4.5.3.

The `__init__` method takes in the ansatz and the initial parameters for the ansatz. An `optimiser` can be passed in as an argument, if so, the optimiser must take the same arguments as `scipy.optimize.minimize`. If not, it is default to `scipy.optimize.minimize`.

The `minimise_eigenvalue` method takes in the Hamiltonian and the number of shots for the simulation. The code snippet below shows how to use the `VQE` class.

```
from Quanthon import VQE
# initialise the parameters with random values
rng = np.random.default_rng(9999)
init_points = rng.random(ansatz.n_params) * 2 * np.pi - np.pi

vqe = VQE(ansatz, init_points)
```

```
        results = vqe.minimise_eigenvalue(qubit_hamiltonian, num_shots
           =10000)
```

4.8.2 AdaptVQE

The `AdaptVQE` class is an implementation of the Adaptive Variational Quantum Eigensolver. To initialise an `AdaptVQE` object,

- an instance of the `QubitAdaptAnsatz` must be passed in;
- an optional boolean parameter `estimate_energy` may be passed in. If set to `True`, the energy is estimated using the `cal_expectation` function from `expectation`, otherwise the analytical values of $\langle\psi|H|\psi\rangle$ is calculated. Default is `True`;
- an `optimiser` may also be passed in, if so, the optimiser must take the same arguments as `scipy.optimize.minimize`. It is default to `scipy.optimize.minimize`;
- upon initialisation, a variable obj_call is initialise

The `run_adapt_circuit` method is called to start the ADAPT circuit, which takes in a qubit Hamiltonian in terms of Pauli operators and their coefficients. Optional parameters include

- `num_shots` for the number of shots in every measurement,
- `grad_eps` default is 10^{-3}. The gradient threshold below which the algorithms stops and is considered converged,
- `method` the classical optimisation method to use,
- `max_iter` the maximum number of ADAPT iterations before the algorithm stops even if convergence gradient is not reached,
- `decompose_exp`, if set to `True`, the exponential of the Pauli strings are decomposed into available gates, if `False` `scipy.linalg.expm` is used to calculate the exponential instead, default is `True`.
- `decompose_method` the method to use for decomposing the exponential, either "staircase" or "inverted staircase", default is "staircase".
- `print_log`, default is `False`, if set to `True`, the energy, operator appended and the gradient at every iteration is printed.
- `opt_log`, default is `False`, sets the value of "disp" in the `scipy.optimize.minimize` method. Returns the optimisation information.

When the `run_adapt_circuit` method is called, it calculates the gradient by calling the `gradient` method for every operator in the pool to select the operator with the largest gradient, then appends it to the circuit of the ansatz and finally calls the `minimise_eigenvalue` method to minimise the `_objective` function which is the energy of the system. If either the maximum iteration is reached or the gradient is below the threshold, the algorithm stops and returns the optimal parameters and the minimised energy.

The `AdaptVQE` class also contain an instance of an `AdaptHistory` object which stores the minimised energy and the operators appended at every ADAPT iteration. It can be accessed via the `hist` member. The code snippet below demonstrates the usage of the `AdaptVQE` class.

```python
        from Quanthon import AdaptVQE
        # initialise the AdaptVQE object with the QubitAdaptAnsatz from
            the previous code block
        adapt_vqe = AdaptVQE(ansatz)
        energy = adapt_vqe.run_adapt_circuit(qubit_hamiltonian,
                                             num_shots=10000,
                                             grad_eps=1e-6,
                                             method="COBYLA",
                                             max_iter=max_iter,
                                             decompose_exp=True,
                                             decompose_method="inverted
                                                 staircase")[1])
        # get a history of the energies at every iteration
        iteration_energy = adapt_vqe.hist.min_energies
```

AdaptHistory

The `AdaptHistory` class stores the following information for every iteration:

- `min_energies` the minimised energy at every iteration,
- `operators_appended` the operators appended at every iteration,
- `max_grads` the maximum gradients for any operator in the ADAPT pool at every iteration.

Optimisers

The classical optimisation is done using the `scipy.optimize.minimize` [**scipy**]. The ones we will be using in this project are:

- `Powell`, the Powell method is minimisation method which performs a series of one-dimensional minimisation along a set of directions to explore the parameter space efficiently until convergence is reached [**powell1964**].
- `COBYLA`, Constrained Optimization BY Linear Approximation is another derivative-free [**powell1994**].
- `SLSQP`, Sequential Least Squares Programming is a gradient-based method [**kraft1988**].
- `BFGS`, Broyden-Fletcher-Goldfarb-Shanno is a quasi-Newton method which approximates the Hessian matrix using the gradients of the objective function [**broyden1970**].
- `L-BFGS-B`, Limited-memory Broyden-Fletcher-Goldfarb-Shanno is a limited-memory version of the BFGS method which also allows bounds [**byrd1995**].
- `Nelder-Mead`, the Nelder-Mead method is a direct search method which does not require the gradient of the objective function [**Nelder1965**].

These methods include both gradient based and gradient free methods. We will later compare their performances in the results section.

Chapter 5

Physical Systems

5.1 Hydrogen Molecule

The hydrogen molecule, (H_2), is the smallest molecule in chemistry. It consists of two hydrogen atoms, each with one electron. In quantum chemistry, the hydrogen molecule is often used as a benchmark for quantum algorithms. Reproducing results of using the VQE to find the ground state energy of the hydrogen molecule is almost a textbook task in quantum computing for electronic structures. The result can also be easily bench marked against traditional quantum chemistry methods, such as Hartree-Fock [ref]. For `Quanthon` , it serves as an example to how to use the software to solve electronic structures problems.

The non-relativistic Hamiltonian for a general molecule in second quantisation is given by

$$H = h_{nuc} + \sum_{pq} h_{pq} a_p^\dagger a_q + \frac{1}{2}\sum_{pqrs} h_{pqrs} a_p^\dagger a_q^\dagger a_s a_r, \tag{5.1}$$

where h_{nuc} is the interaction between nuclei and electrons, and h_{pq} and h_{pqrs} are the one and two electron integrals respectively.

$$\begin{aligned} h_{pq} &= \int d\sigma \varphi_p^*(\sigma)\left(-\frac{\nabla_{\vec{r}}^2}{2} - \sum_i \frac{Z_i}{\left|\vec{R}_i - \vec{r}\right|}\right)\varphi_q(\sigma), \\ h_{pqrs} &= \int d\sigma_1 d\sigma_2 \frac{\varphi_p^*(\sigma_1)\,\varphi_q^*(\sigma_2)\,\varphi_s(\sigma_1)\,\varphi_r(\sigma_2)}{|\vec{r}_1 - \vec{r}_2|}, \\ h_{nuc} &= \frac{1}{2}\sum_{i\neq j}\frac{Z_i Z_j}{\left|\vec{R}_i - \vec{R}_j\right|}, \end{aligned} \tag{5.2}$$

where Z_i is the charge of the nuclei and $\vec{R}$ and $\vec{r}$ are the spatial coordinate of the nuclei and electron respectively. The function φ is the one electron functions, often called spin orbitals in quantum chemistry, are usually obtained from methods such as the Hartree Fock [**Romero2018**]. Fortunately, for the hydrogen molecule and many others, these integrals have been pre-calculated and we will be able to obtain them by using Packages such as PySCF [**Sun2018**].

We will use the minimal basis set, namely the STO-3G basis set, which consists of three Gaussian functions for each atomic orbital [ref]. The choice of the basis set is to minimise the number of qubits required to represent the system since it is the accuracy of our VQE algorithms that is our main concern rather than capturing all physics of the system.

In this basis then we obtained the integrals for the hydrogen molecule to be as shown in Table 5.1 and Table 5.2. With molecule geometry of

$$\begin{aligned} \vec{R}_1 &= (0,0,0), \\ \vec{R}_2 &= (0,0,0.735), \end{aligned} \tag{5.3}$$

in units of angstrom (Å).

Table 5.1: One-electron integral values for the STO-3G basis set.

Indices	Integral values
$h_{00} = h_{11}$	-1.252477
$h_{22} = h_{33}$	-0.47189601

Table 5.2: Two-electron integral values for the STO-3G basis set.

Indices	Integral values
$h_{0000} = h_{0220} = h_{2002} = h_{2222}$	0.33785508
$h_{0011} = h_{0101} = h_{0231} = h_{0321} = h_{1010} = h_{1100} = h_{1230} = h_{1320}$	0.09046560
$h_{2013} = h_{2103} = h_{2233} = h_{2323} = h_{3012} = h_{3102} = h_{3232} = h_{3322}$	0.09046560
$h_{0110} = h_{0330} = h_{1001} = h_{1221} = h_{2332} = h_{3003} = h_{3223}$	0.33229087
$h_{1111} = h_{1331} = h_{3113} = h_{3333}$	0.34928686

With these information we could now perform the Jordan-Wigner transform implemented in Subsection 4.5.1. The resulting qubit Hamiltonian is

Table 5.3: Qubit Hamiltonian after Jordan-Wigner transform.

Pauli String	**Coefficients**
IIII	-0.81054798
IIIZ	-0.22575349
IIZI	0.17218393
IIZZ	0.12091263
IZII	-0.22575349
IZIZ	0.17464343
IZZI	0.16614543
XXXX	0.04523280
XXYY	0.04523280
YYXX	0.04523280
YYYY	0.04523280
ZIII	0.17218393
ZIIZ	0.16614543
ZIZI	0.16892754
ZZII	0.12091263

The distance between the two hydrogen atoms are chosen to be 0.735 angstrom, which is the equilibrium bond length of the hydrogen molecule. We will repeat the process for different bond lengths to obtain the dissociation curve of a molecule as well as observe the behaviour of the VQEs for different bond lengths. Finally, since the VQE is only used to compute the electronic part of the Hamiltonian, the nuclear part will be computed separately and added to the total energies.

5.2 Lipkin-Meshkov-Glick Model

The Lipkin-Meshkov-Glick (LMG) model was first introduced to be a model with simple analytical solutions and yet not trivial to be solved [**lipkin1965**]. The model is a simple model of a system of N fermions with two levels, denoted by $|1\rangle$ and $|2\rangle$. Today it serves as a test bed for more advanced algorithm since the analytical solutions are well known. The LMG model is a two-level system with N fold degeneracy for N particles in the system. The upper shells are associated with quantum number $\sigma = +1$ and the lower shells are associated with quantum number $\sigma = -1$ as illustrated .

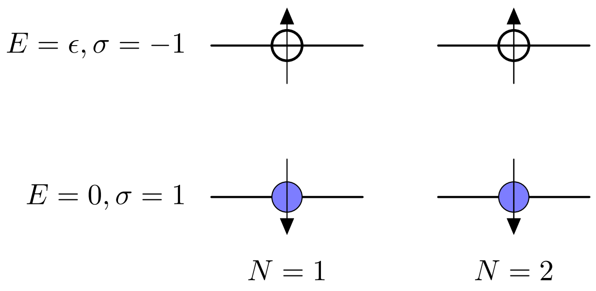


Figure 5.1: Illustration of the LMG model with 2 doubly degenerate single particle states and 2 particles.

If $\alpha^\dagger_{p\sigma}$ is the creation operator for a particle in the p state on the σ level, the Hamiltonian of the LMG model is given by

$$\begin{aligned} H_0 &= \varepsilon J_z, \\ H_1 &= \frac{1}{2} V \sum_{p,p',\sigma} a^\dagger_{p\sigma} a^\dagger_{p'\sigma} a_{p'-\sigma} a_{p-\sigma}, \\ H_2 &= \frac{1}{2} W \sum_{p,p',\sigma} a^\dagger_{p\sigma} a^\dagger_{p'-\sigma} a_{p'\sigma} a_{p-\sigma}, \end{aligned} \tag{5.4}$$

where H_0 is a single particle term and H_1 and H_2 are the interaction terms. The total Hamiltonian is the sum of all three components,

$$H = H_0 + H_1 + H_2.$$

Define quasi-spin operators $J_z, J_\pm$ as

$$\begin{aligned} J_z &= \frac{1}{2} \sum_{p,\sigma} a^\dagger_{p\sigma} a_{p\sigma}, \\ J_+ &= \sum_p a^\dagger_{p\sigma} a_{p-\sigma}, \\ J_- &= \sum_p a^\dagger_{p-\sigma} a_{p\sigma}. \end{aligned} \tag{5.5}$$

The Hamiltonian can be rewritten in terms of quasi-spin operators,

$$\begin{aligned}
H_0 &= \varepsilon J_z, \\
H_1 &= \frac{1}{2} V \left(J_+^2 + J_-^2 \right) \\
H_2 &= \frac{1}{2} W \left(-N + J_+ J_- + J_- J_+ \right)
\end{aligned} \tag{5.6}$$

The quasi-spin operators satisfy the following relations

$$J_+ |J, J_z\rangle = \sqrt{J(J+1) - J_z(J_z+1)} |J, J_z + 1\rangle, \tag{5.7}$$

$$J_- |J, J_z\rangle = \sqrt{J(J+1) - J_z(J_z-1)} |J, J_z - 1\rangle. \tag{5.8}$$

Instead of using the Jordan-Wigner transform to convert the second quantised Hamiltonian into a qubit Hamiltonian, these properties of the LMG model allows us to rewrite the Hamiltonian directly into Pauli matrices. We will discuss two cases for $J = 1$ and $J = 2$, then a discussion of the general case.

5.2.1 Case $J = 1$

Acting J_+ and J_- on all possible states to get the matrix elements:

$$J_+ |1, -1\rangle = \sqrt{2} |1, 0\rangle$$

$$J_+ |1, 0\rangle = \sqrt{2} |1, 1\rangle$$

$$J_+ |1, 1\rangle = 0$$

We can obtain a set of similar relations for J_-. Hence the matrix representation of the operators are

$$J_z = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad J_+^2 = \begin{bmatrix} 2 & 2 & 0 \\ 2 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad J_-^2 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 2 \\ 0 & 2 & 2 \end{bmatrix} \tag{5.9}$$

Substituting the matrix representations of the operators into Equation (5.6), we obtain

$$H_{J=1} = \begin{bmatrix} -\epsilon & 0 & v \\ 0 & W & 0 \\ v & 0 & \epsilon \end{bmatrix} \tag{5.10}$$

We now want to rewrite the Hamiltonian in terms of Pauli matrices. We set $W = 0$, then

$$H = H_0 + VH_1$$

$$\begin{aligned}
J_z &= \sum_n j_z^{(n)} = \frac{1}{2}(Z_1 + Z_2) \\
\implies H_0 &= \epsilon J_z = \frac{\epsilon}{2}(Z_1 + Z_2)
\end{aligned} \tag{5.11}$$

$$
\begin{aligned}
H_1 &= \frac{1}{2}V(J_+^2 + J_-^2) \\
&= \frac{1}{2}V\left[(\sum_n j_+^{(n)})^2 + (\sum_n j_-^{(n)})^2\right] \\
&=\frac{1}{2}V\left(\sum_{nm}\left(j_+^{(n)}j_+^{(m)} + j_-^{(n)}j_-^{(m)}\right)\right) \\
&= \frac{1}{2}V\sum_{nm}\left(j_x+ij_y\right)^{(n)}\left(j_x+ij_y\right)^{(m)} + \left(j_x-ij_y\right)^{(n)}\left(j_x-ij_y\right)^{(m)} \\
&= \frac{1}{2}V\sum_{nm}(j_x^{(n)}j_x^{(m)} + ij_y^{(n)}j_x^{(m)} - j_y^{(n)}j_y^{(n)} + j_x^{(n)}j_x^{m} - ij_y^{n}j_x^{(m)} - j_y^{(n)}j_y^{(m)})
\end{aligned} \tag{5.12}
$$

Since

$$
j_x = \frac{X}{2}, j_y = \frac{Y}{2}, j_z = \frac{Z}{2}
$$

Equation (5.11) and (5.12) becomes

$$
\begin{aligned}
&v\sum_{nm}\left(j_x^{(n)}j_x^{(m)} - j_y^{(n)}j_y^{(m)}\right) \\
&= 2v\sum_{n<m}\left(\frac{X_n}{2}\frac{X_m}{2} - \frac{Y_n}{2}\frac{Y_m}{2}\right)
\end{aligned}
$$

$$
\implies H_1 = \frac{1}{2}V(X_1\otimes X_2 - Y_1\otimes Y_2) \tag{5.13}
$$

Combining H_0 and H_1

$$
H = \frac{\epsilon}{2}(Z_1+Z_2) + \frac{1}{2}V(X_1\otimes X_2 - Y_1\otimes Y_2) \tag{5.14}
$$

5.2.2 Case $J = 2$

Using again Equation (5.7), we obtain the Hamiltonian matrix for $J = 2$.

$$
H_{J=2} = \begin{pmatrix}
-2\varepsilon & 0 & \sqrt{6}V & 0 & 0 \\
0 & -\varepsilon+3W & 0 & 3V & 0 \\
\sqrt{6}V & 0 & 4W & 0 & \sqrt{6}V \\
0 & 3V & 0 & \varepsilon+3W & 0 \\
0 & 0 & \sqrt{6}V & 0 & 2\varepsilon
\end{pmatrix} \tag{5.15}
$$

Since the Hamiltonian in terms of the quasi-spin operators are general regardless of the values of J , we can rewrite the Hamiltonian in terms of the quasi-spin operators directly for the $J = 2$ case. Note that the H_0 is a one body term, we take all linearly combination of the Z and I, the identity operator with one Z, the X and Y terms both act on two bodies, so we need to take all combination of XX (or YY) and II. The tensor product $\otimes$ sign is omitted for simplicity.

$$
\begin{aligned}
H = \ &\epsilon\,(ZIII + IZII + IIZI + IIIZ) \\
&+ \frac{V}{2}(XXII + XIXI + XIIX + IXXI + IXIX + IIXX) \\
&- \frac{V}{2}(YYII + YIYI + YIIY + IYYI + IYIY + IIYY)
\end{aligned} \tag{5.16}
$$

Again, rewriting the Hamiltonian in terms of Pauli matrices allows us to perform quantum computation with them.

We could also rewrite H_2 in terms of the Pauli matrices. Then second term can be rewritten in similar ways to H_1

$$\implies H_2 = \frac{1}{2}W(XXII + \cdots IIXX + YYII \cdots IIYY).$$

Setting $W = 0$, the Hamiltonian for the system is Equation (5.16).

5.2.3 Level Mapping

Level mapping uses the symmetry of the LMG Hamiltonian to reduces the dimension (number of qubits) required for the mapping. One way, perhaps the more natural way, to map the fermions is by using one qubit to represent a state (n, σ) where n is the particle slot and σ is the level (either spin up or down). This is the same idea as the Jordan-Wigner mapping, which is wasteful in this case due to the symmetry of the LMG model. In this mapping it requires $2N$ qubits to represent a system of N particles since there are 2 levels.
However, we know also that the LMG model do not permit the shift between particle slots, hence given an n, it's not possible that both σ's to be occupied at the same time. By taking advantage of this situation, we could consider the "level mapping", where we associate a qubit with a doublet, with

$$|0\rangle \longleftrightarrow |n, -1\rangle$$
$$|1\rangle \longleftrightarrow |n, +1\rangle .$$

This allows us to map two states to one qubit, requiring only N qubits for a system of N particles [**Cervia21**].

5.2.4 Dimensionality Reduction for Pauli Decomposition

In the $J = 1$ case, Equation (5.10) has a redundant dimension in the $W = 0$ case since the second row and column contain only 0 entries. This means we could effective reduce the dimension of the matrix to 2×2, as shown in Equation (5.17) which means only one qubit is required to represent the system instead of 2.

$$H_{J=1} = \begin{bmatrix} -\epsilon & v \\ v & \epsilon \end{bmatrix} \tag{5.17}$$

For the $J = 2$ case, whilst by setting $W = 0$ we do not automatically reduce the dimension of the matrix, through exchanging rows and columns we could rewrite the matrix in a way that the Hamiltonian is block diagonal. Where the top left block of the Hamiltonian represents the cases where the values of J_z are even and the bottom right matrix represents the cases where the values of J_z are odd. This means that we could effectively write the Hamiltonian matrix into two non-interacting matrices.

$$\begin{pmatrix} -2\epsilon & \sqrt{6}V & 0 & 0 & 0 \\ \sqrt{6}V & 4W & \sqrt{6}V & 0 & 0 \\ 0 & \sqrt{6}V & 2\epsilon & 0 & 0 \\ 0 & 0 & 0 & -\epsilon + 3W & 3V \\ 0 & 0 & 0 & 3V & \epsilon + 3W \end{pmatrix} \tag{5.18}$$

We will compare in Section 6.5 the results of the LMG model using level mapping with Pauli Decomposition.

5.3 Pairing Model

The pairing model is a simple model of a system of fermions with pairing interaction, consists of N doubly degenerate energy levels with equal spacing. It reduces the computation complexity while maintaining a grasp of the essential physics of the system. The theory of superconductivity in which electrons form cooper pairs is described by the pairing interaction known as the BCS model [**Bardeen1957**] [**Cooper1959**].

The pairing Hamiltonian is given by

$$\hat{H}_0 = \xi \sum_{p,\sigma} (p-1) \hat{a}^\dagger_{p\sigma} \hat{a}_{p\sigma} \tag{5.19}$$

$$\hat{V} = -\frac{1}{2} g \sum_{pq} a^\dagger_{p+} a^\dagger_{p-} a_{q-} a_{q+} \tag{5.20}$$

and the full Hamiltonian is

$$H = H_0 + V.$$

We will select a case of the pairing model with 4 single particle states and 4 particles. Without any constrain the Hilbert space has size $\binom{8}{4} = 70$, which would require at least 7 qubits to represent the system, which is too large to simulate locally. However, the pairing Hamiltonian allows us to reduce the dimension of the Hilbert space by restricting the system to have no broken pairs as in the middle figure of Figure 5.2.

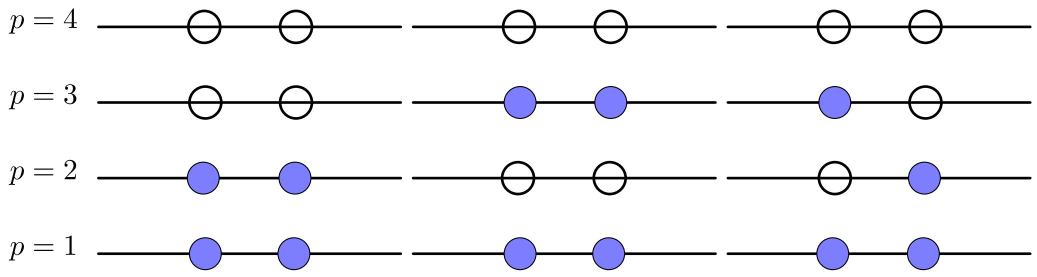


Figure 5.2: Illustration of the paring model with 4 doubly degenerate single particle states and 4 particles. Left shows all the particles are in the lowest lying states. The middle figure shows a doubly excited state where no pairs are broken. The right figure shows a state with a singly excited particle where a pair is broken.

Without loss of generality, we set $\xi = 1$ and vary the values of g, the pairing strength. Consider the spin project operator

$$\hat{S}_z = \frac{1}{2} \sum_{p,\sigma} \sigma \hat{a}^\dagger_{p\sigma} \hat{a}_{p\sigma} \tag{5.21}$$

$$\hat{S}^2 = \hat{S}_z^2 + \frac{1}{2}\left(\hat{S}_+ \hat{S}_- + \hat{S}_- \hat{S}_+\right) \tag{5.22}$$

$$\hat{S}_\pm = \sum_p \hat{a}^\dagger_{p\pm} \hat{a}_{p\mp}. \tag{5.23}$$

The pair creation and annihilation operators are defined as

$$\hat{P}^+_p = \hat{a}^\dagger_{p+} \hat{a}^\dagger_{p-}, \quad \hat{P}^-_p = \hat{a}_{p-} \hat{a}_{p+}. \tag{5.24}$$

The pairing Hamiltonian can be rewritten as

$$\hat{H} = \sum_{p\sigma}(p-1)a^\dagger_{p\sigma}a_{p\sigma} - \frac{1}{2}g\sum_{pq}\hat{P}^+_p\hat{P}^-_q. \tag{5.25}$$

To set up the matrix Hamiltonian for where no pair is broken we need to set up the basis for the system. We include all possible combinations for the 4 particles to occupy the 4 doubly degenerated states without breaking any pairs as the basis. Such that

$$\begin{aligned}
|\phi_0\rangle &= |1_+, 1_-, 2_+, 2_=\rangle = \hat{a}^\dagger_{2+}\hat{a}^\dagger_{2-}\hat{a}^\dagger_{1+}\hat{a}^\dagger_{1-} = \hat{P}^+_2\hat{P}^+_1|0\rangle \\
|\phi_1\rangle &= |1_+, 1_-, 3_+, 3_-\rangle = \hat{P}^+_3\hat{P}^+_1|0\rangle \\
|\phi_2\rangle &= |1_+, 1_-, 4_+, 4_-\rangle = \hat{P}^+_4\hat{P}^+_1|0\rangle \\
|\phi_3\rangle &= |2_+, 2_-, 3_+, 3_-\rangle = \hat{P}^+_3\hat{P}^+_2|0\rangle \\
|\phi_4\rangle &= |2_+, 2_-, 4_+, 4_-\rangle = \hat{P}^+_4\hat{P}^+_2|0\rangle \\
|\phi_5\rangle &= |3_+, 3_-, 4_+, 4_-\rangle = \hat{P}^+_4\hat{P}^+_3|0\rangle.
\end{aligned} \tag{5.26}$$

Let

$$|\Phi_0\rangle = \begin{pmatrix}1\\0\\0\\0\\0\\0\end{pmatrix}, \quad |\Phi_1\rangle = \begin{pmatrix}0\\1\\0\\0\\0\\0\end{pmatrix}, \quad \cdots \quad |\Phi_5\rangle = \begin{pmatrix}0\\0\\0\\0\\0\\1\end{pmatrix},$$

be the basis for the system, and compute the matrix element $\langle\phi_i|\hat{H}|\phi_j\rangle$ using Equation (5.24). We obtain that the matrix representation of the Hamiltonian

$$\hat{H} = \begin{pmatrix}
2-g & -g/2 & -g/2 & -g/2 & -g/2 & 0 \\
-g/2 & 4-g & -g/2 & -g/2 & 0 & -g/2 \\
-g/2 & -g/2 & 6-g & 0 & -g/2 & -g/2 \\
-g/2 & -g/2 & 0 & 6-g & -g/2 & -g/2 \\
-g/2 & 0 & -g/2 & -g/2 & 8-g & -g/2 \\
0 & -g/2 & -g/2 & -g/2 & -g/2 & 10-g
\end{pmatrix}. \tag{5.27}$$

We will encode this Hamiltonian using Pauli Decomposition into qubit Hamiltonian for the VQEs. The Hamiltonian has 18 terms.

The qubit Hamiltonian is given in Table **??**.

5.4 Deuteron Model

The deuteron model is a bound state of a proton and a neutron, which is surprisingly stable since neutrons are famously not [ref]. We follow the derivation of the deuteron Hamiltonian as [**Dumitrescu2018**] [**Binder2016**] [**Bansal2017**] for our simulation, where a discrete variable representation in the harmonic oscillator basis is used. The Hamiltonian is given as

$$H_N = \sum_{n,n'=0}^{N-1}\langle n'|(T+V)|n\rangle\, a^\dagger_{n'}a_n. \tag{5.28}$$

where n is the harmonic oscillator quantum number, $a^\dagger_n, a_n$ are the creation and annihilation operators for the deuteron in the harmonic oscillator s-wave state $|n\rangle$, T is the kinetic energy

Pauli String	**Coefficient**
III	28.5
IIX	-0.25
IIZ	-0.5
IXI	-0.25
IXX	-0.25
IZI	-23.5
IZX	-0.25
IZZ	-0.5
XII	-0.25
XXI	-0.25
XXX	-0.25
XZI	-0.25
YYI	-0.25
YYX	-0.25
ZII	-25
ZXI	-0.25
ZXX	-0.25
ZZI	22

Table 5.4: Pauli strings and their coefficients.

operator and V is the potential energy operator. The kinetic energy operator T and potential energy operator V are given by

$$\begin{aligned}\langle n'|T|n\rangle =&\frac{\hbar\omega}{2}\left[(2n+3/2)\delta_n^{n'} - \sqrt{n(n+1/2)}\delta_n^{n'+1} - \sqrt{(n+1)(n+3/2)}\delta_n^{n'-1}\right], \\ \langle n'|V|n\rangle =&V_0\delta_n^0\delta_n^{n'},\end{aligned} \tag{5.29}$$

where $V_0 = -5.68658111$ MeV and $\hbar\omega = 7$ MeV.

The Jordan-Wigner transform was used in [**Dumitrescu2018**] to map the Deuteron Hamiltonian to qubit Hamiltonian, but observing Equation (5.28) it does not have any two-body term, which means transforming it with Jordan-Wigner transformation is wasteful, and rewriting it into a Hamiltonian matrix then use the Pauli decomposition saves the number of qubits required to represent the system, which is a huge advantage for any algorithm running on NISQ devices, especially since the basis dimension N can increase indefinitely. Since we are able to save on the number of qubits we will push the basis dimension N to higher values than the $N = 3$ in the original paper.

5.5 VQE Utils

The `vqe_utils.py` file contains two essential function which assists us to simulate the above system using the quantum computing library `Quanthon`. The `run_hw` function uses one of the two native fixed form ansatz, the `HardwareEfficientAnsatz` or `RyAnsatz`. It takes in the following arguments

- `h` : The qubit Hamiltonian of the system.

- `method` : The classical optimiser to use.
- `rep` : The number of repetitions of the ansatz. Default is 2.
- `estimate_energy` : If `True` the energy is estimated using the `cal_expectation` function, if `False` the energy is calculated exactly. Default is `True`.
- `n_shots` : The number of shots used for measurements. Default is 10000.
- `ansatz_type` : The type of ansatz to use, either `hw` or `ry`. Default is `hw`.
- `init_state` : The initial state of the system. If `None` the initial state is the equal superposition state $|+^{\otimes n}\rangle$. Default is `None`.
- `opt_log` : If `True` the optimisation log is printed. Sets the values for "disp" in the optimiser.Default is `False`.

The function returns the energy of the system and the final state of the qubits.

The other function `run_adapt` allows us to run the ADAPT-VQE with the `QubitAdaptAnsatz`. It takes in the following arguments.

- `h` : The qubit Hamiltonian of the system.
- `max_iter` : The maximum number of ADAPT iterations allowed. Default is 100.
- `pool` : The ADAPT operator pool. Default is "V".
- `num_shots` : The number of shots used for measurements. Default is 10000.
- `method` : The classical optimiser to use. Default is `COBYLA`.
- `decompose_exp` : If `True` the exponential of the Hamiltonian is decomposed into hardware friend gates, if not, `scipy.linalg.expm` is used. Default is `True`.
- `decompose_method` : The method to use for the decomposition of the exponentials. Default is "inverted staircase".
- `init_state` : The initial state of the system. If `None` the initial state is the equal superposition state $|+^{\otimes n}\rangle$. Default is `None`.
- `print_log` : If `True` the log of the ADAPT process is printed. Default is `False`.
- `opt_log` : If `True` the optimisation log is printed. Sets the values for "disp" in the optimiser.Default is `False`.

Part III

Results and Discussion

Chapter 6

Results

In this chapter we presents results from both exact energy simulation (with no shot noise) and ideal simulation (with shot noise) for the Hydrogen molecule, LMG model, Pairing model and the Deuteron model. The results are grouped based on the models they belong to and the analysis of properties for the qubit-ADAPT-VQE and the fixed-form ansatz are distributed into different sections. When a result is produced "without shot noise", the analytical expression for the energy is calculated. This is equivalent taking `n_shots` $\to \infty$. The optimisers are methods from`scipy.optimize.minimize`. If the exponential decomposition method is not mention explicitly, the inverted staircase is used. If the initialisation of a state is not mentioned explicitly, it is in initialised in the maximally superposed state.

6.1 Expected Error

Even with our simulations being ideal, the energy estimations are not exact. Therefore, the best results one aims to achieve is the one whose energy error is at its theoretical minimum, i.e. only comes from shot noise. As described in Section 4.2.2, the error obtained from one measurement is given by

$$\epsilon = \sqrt{\frac{1}{N}} \tag{6.1}$$

where N is the number of shots.

For a qubit Hamiltonian with t terms, we define the expected error E_ϵ error is then

$$E_\epsilon = t\epsilon$$

.

The maximum number of terms of Pauli strings a qubit Hamiltonian contains for a n qubit system is 4^n. For a 3 qubit system, the maximum number of terms is 64 and 256 for 4 qubits. For example, for our common choice 10^5 shots, the minimum expected error for a three qubit system with all 64 terms will be

$$\times\sqrt{\frac{1}{64 \times 10^5}} \approx 0.025 \sim \mathcal{O}(10^{-2}).$$

Although the system usually does not contain all the terms, so the this assessment will be conduced for every system separately as a reference point for the results.

6.2 Run Time

The time taken to run each simulation is highly related to the number of shots, the size of the Hamiltonian and the number of objective functions called during the classical optimisation process. The time taken for the measurement scales linearly with the number of shots, as can be seen in Figure 6.1.

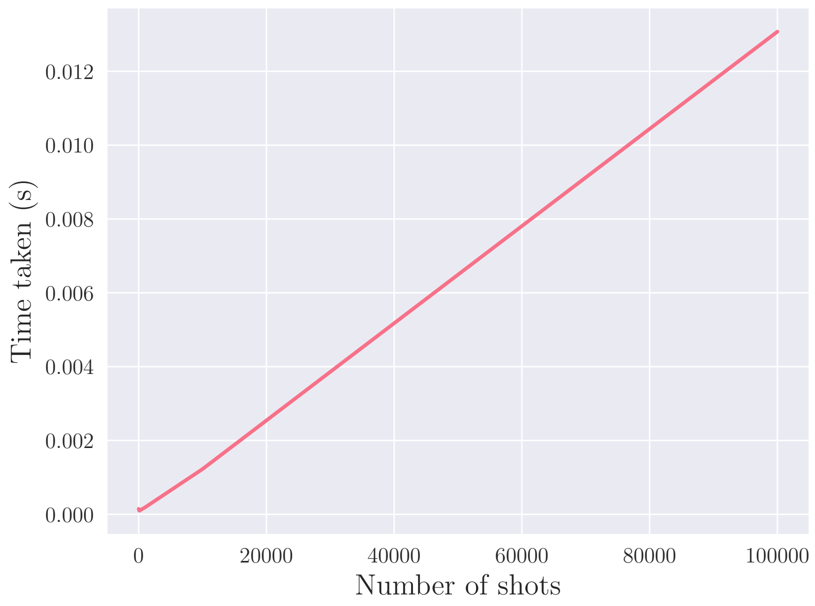


Figure 6.1: Time taken in seconds for the measurement process as a function of the number of shots.

The time taken for every measurement with 10^5 shot is around $0.008374s$. The total time taken for every calculation should also depend on the number of function calls and the length of the qubit Hamiltonian.

6.3 State Initialisation for ADAPT-VQE

According to [**tang2021**], the pools V and G should be complete, i.e. any real state $|\psi\rangle$ can be rotated to another state $|\phi\rangle$ with

$$|\phi\rangle = \prod_k e^{\theta V_k} |\psi\rangle . \tag{6.2}$$

However, the ADAPT stops when the gradient is 0 or below or certain threshold. Even after removing the even Pauli strings, i.e. Pauli string with even number of Y operators, the operator gradient can still vanish for some states $|\psi\rangle$. Most noticeably, the $|0\dots0\rangle$ state. We will provide an example for where all the gradient of the operators vanish, and the ADAPT process stops after one iteration without converging to the correct state.

For the case of three qubits, the complete V pool is $\{V\} = \{iYZZ, iIYZ, iIIY, iIYI\}$. In matrix notation that is

$$iYZZ = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{6.3}$$

Initialising the state

$$|\psi\rangle = |000\rangle$$

for the operator selection process.

We calculate the gradient of each operator in the pool for an arbitrary Hamiltonian with real coefficients, which is true when time reversal symmetry is preserved. The Hamiltonian of choice is

$$H = 2IIZ - 0.5IXX + 0.5IYY + 2ZZI \tag{6.4}$$

in matrix notation that is

$$\begin{pmatrix} 4 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & -4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & -4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \end{pmatrix} \tag{6.5}$$

The gradient is calculated using Equation (3.32). The commutator $[H, A]$ with $A = iYZZ$ is then

$$
\begin{aligned}
[H, A] = HA - AH &= \begin{pmatrix} 4 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & -4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & -4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \end{pmatrix} \\
&- \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 4 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & -4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & -4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \end{pmatrix} \\
&= \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}
\end{aligned}
$$

and

$$
\begin{aligned}
\langle\psi|[H, A]|\psi\rangle &= \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 0 & 0 & -1 & 4 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & -4 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 4 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & -4 \\ 4 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & -4 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 4 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -4 & 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} && (6.6) \\
&= \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 0 \\ -1 \\ 4 \\ 0 \\ 0 \\ 0 \end{pmatrix} && (6.7) \\
&= 0 && (6.8)
\end{aligned}
$$

In Appendix B we will show that this is true for all the rest of the operators in the pool when the $|000\rangle$ is obviously not the ground state for the system, which is supposed to have energy -4.236. The initialisation of the state is therefore crucial. We found that by starting in the maximally superposed state, $|+\rangle^{\otimes n}$, the gradient of the operators are non-zero, and the system does converge to the ground state eventually. This will therefore be the choice of initial state for the ADAPT-VQE from now on.

6.4 Hydrogen Molecule

While producing results for the hydrogen molecule, we found that initialising the state in the Hartree-Fock state ($\langle 1100\rangle$in this case) results in the 0 gradient problem as described in Section 6.3 and Appendix B. Indeed, all terms of the qubit Hamiltonian shown in Table 5.3 do not contribute to the gradient when the state is in the $|0000\rangle$ or the $|0000\rangle$ state.

The qubit Hamiltonian contains 15 terms. The expected error for 10^5 shots is 0.00316. The expected error for the hydrogen molecule is therefore 0.012.

6.4.1 Exact Energy Simulation

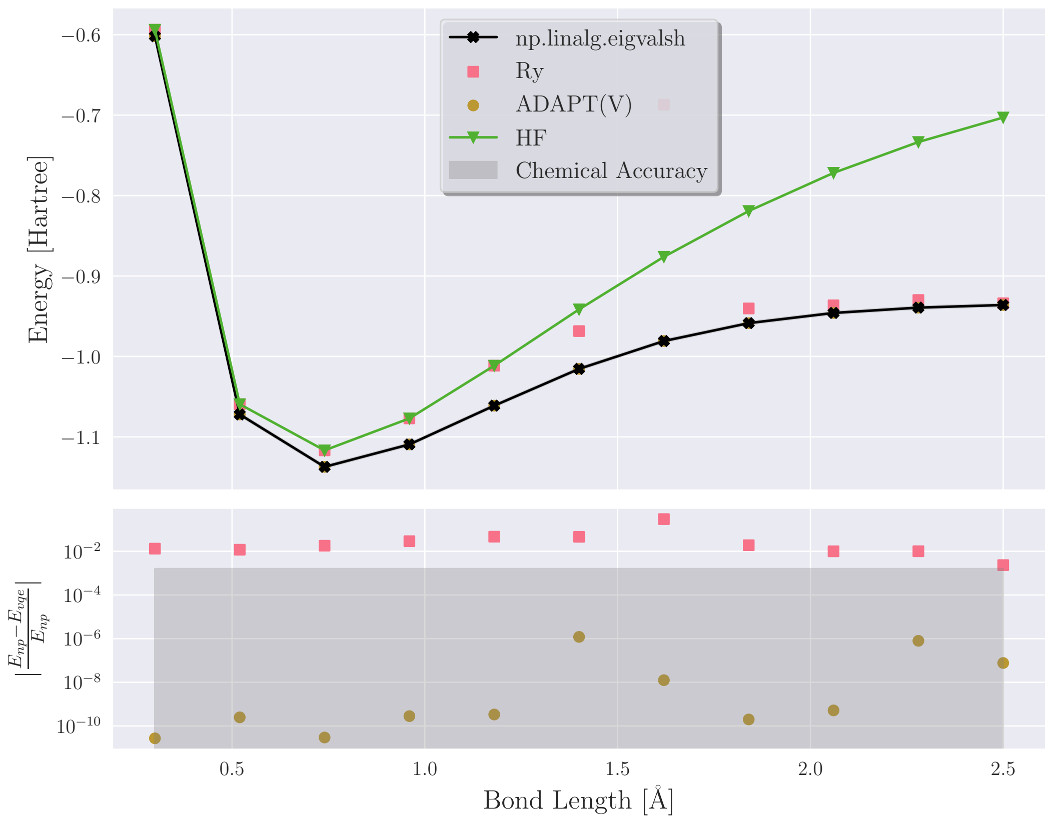


Figure 6.2: The Hydrogen molecule with **exact energy simulation**, showing comparison between VQE with 3 rep and ADAPT-VQE with the V pool and maximum 30 ADAPT iterations. Both minimised using the `BFGS` method. The VQE was initialised in the HF state and the ADAPT-VQE was initialised in a random real state.

We could see for the exact energy simulation the ADAPT-VQE outperforms the VQE and

Hatree-Fock by large margin, and was able to converge to the exact diagonalisation with an error within orders of 10^{-6} for both small and large bond length, also achieving chemical accuracy. The Ry ansatz was not able to converge to the true ground state for small bond length, rather it converges to the HF state. This might be related to the Hartree-Fock initialisation. For large bond length where the Hartree-Fock energy is a poor estimation of the ground state, both VQEs has better results than the Hartree-Fock energy. We also note that the performance of the ADAPT-VQE is not affected by the bond length as the error stays the same, suggesting that the qubit-ADAPT ansatz is exact and is able to represent ground state with different interactions.

6.4.2 Ideal Simulation

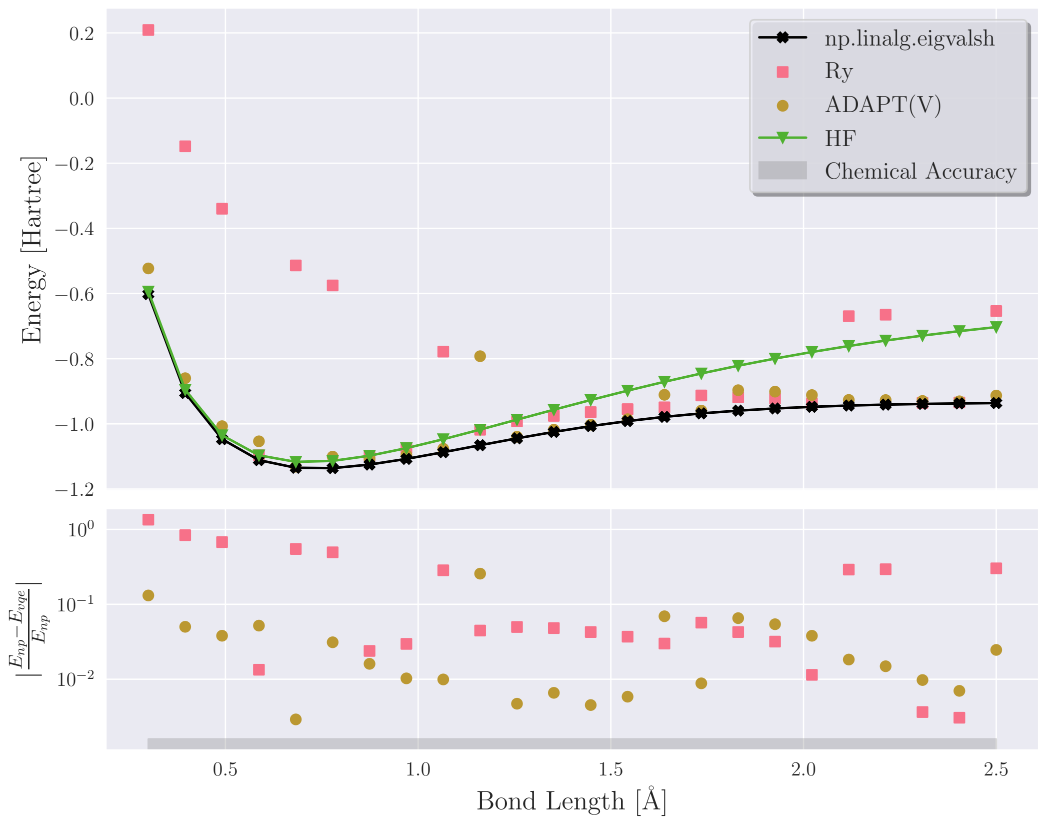


Figure 6.3: The hydrogen molecule with **ideal simulation** with 100000 shots for a maximum of 30 ADAPT iterations. The ADAPT-VQEs were optimised with the `COBYLA` method and the VQE with the `Powell` method. The exponential was decomposed using the **inverted Staircase** algorithm.

Figure 6.3 shows the results for ideal simulation. Again, ADAPT outperformed the VQE but the result for both were underwhelming and were not chemically accurate. It is still obvious that the ADAPT-VQE is more consistent in terms of performance when it comes to bond length,and the normal VQE with Ry ansatz perform poorly for both low and high bond length. Compare to the expected error with this shot noise 0.012 , only few points between $1 - 1.5$Å with the use of the ADAPT-VQE were able to achieve. Clearly, the simple shot noise affects the convergence of the ADAPT-VQE greatly. Even with the ADAPT gradient being calculated analytical, the impact of the shot noise is still significant. Our hypothesis is that the optimisation for the VQE subroutine at every ADAPT iteration might not always converge, causing the state to be in a different state than the state with the optimal parameters for the current iteration. One suggestion to improve the result is to group commuting operators to be measured together to reduce the number of state preparation required.

To test our hypothesis, we ran the system with the same parameters except increasing the maximum iteration to 60. The result for this is shown in Figure C.3. While this did not improve our results it helps us understand the problem better. For all the values, the energy was decreasing with every iteration initially, but stopped after 10 iterations. For the case of 0.3Å, the error is below the expected error for the number of shots we use, as we can see from the fluctuation of the error after the convergence. Results for other bond lengths however was not able to reach the exact ground state as while more operators were being appended, the error was not decreasing to below the expected error. Meanwhile, the gradient is not small enough for the ADAPT-VQE to stop, which may suggest that with shot noise, the operator gradient might have trouble selecting the operator which changes the state the most. This also tells us that using expected error is a good metric to evaluate the performance of an ideal simulation. This, incidentally, also result in a large increase of function calls, approximately 15000, where ~ 1000 function calls were made for the 15 iteration case and ~ 5000 for the 30 iteration case. This is a sign that the optimisers struggles as the number of parameters increases, which is unsurprising.

6.5 Lipkin-Meshkov-Glick Model

In section we showcase results obtained from both exact energy and ideal simulation for the LMG model, for both the $J = 1$ and $J = 2$ cases. We will assume $W = 0$ throughout this section.

Both the rewritten and Pauli decomposed Hamiltonian for $J = 1$ has 4 terms, and $J = 2$ has 16 terms. The expected error for these case with 10^5 shots are 0.006 and 0.013 respectively.

6.5.1 Exact Energy Simulation

Figure 6.4 and Figure 6.5 shows the comparison between VQE and ADAPT-VQE for the LMG model with $J = 1$ and $J = 2$ for the **exact energy simulation** respectively. We found that with *rep* $= 1$, the VQE algorithm was able to converge, but an attempt to improve the result further by adding another layer of gates caused the algorithm to fail to converge completely for the case of rewritten Hamiltonian as shown in Figure C.4. For $J = 1$, apart from a few points where one method did not converge, the convergence was decent for all the methods, with errors in orders of 10^{-8}. For the $J = 2$ case the ADAPT with Pauli decomposed energy produced the best results with errors orders of 10^{-8} while other methods had errors between 10^{-2} and 10^{-1}. In both cases, the ADAPT performed better than the VQE respectively by being more stable for all interaction strength and lower error.

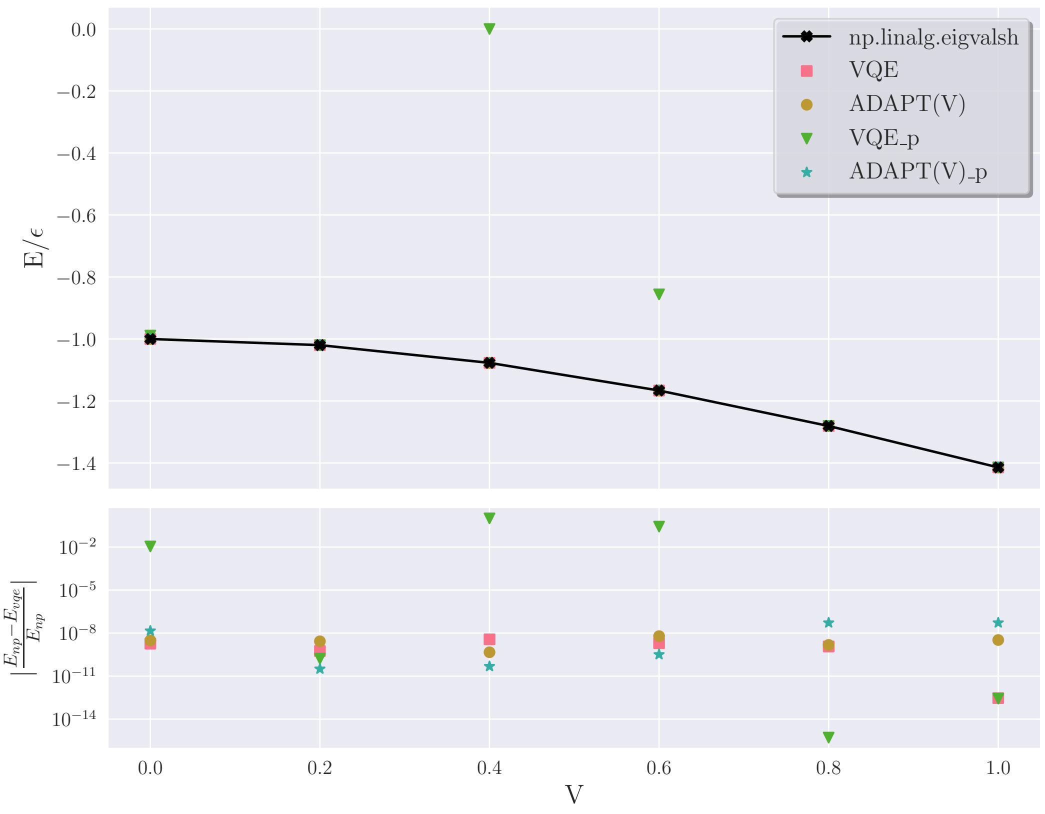


Figure 6.4: The LMG model with $J = 1$ with **exact energy simulation**, showing comparison between VQE with 1 rep and ADAPT-VQE with the V pool and maximum 12 ADAPT iterations. The label with *Pauli* means that the Hamiltonian was mapped using Pauli decomposition, otherwise Equation (5.11) was used. Both optimised with `BFGS` method.

6.5.2 Ideal Simulation

Figure 6.6 and 6.7 shows the results for ideal simulations again for both $J = 1, 2$ as well both mapping.

6.5.3 Circuit Properties

Table 6.1 shows the comparison between the fixed-form ansatzes and the qubit-ADAPTs for the $J = 1$ case with $V = 1$. Both qubit-ADAPTs have more parameters and gates, but the number of energy function evaluations is also less for qubit-ADAPT is fewer than the HardwareEfficientAnsatz. Pauli-decomposing the Hamiltonian also reduces the resource required to solve the problem. As the hardware efficient ansatz have much fewer gates and CNOTgates, it is expected that there are limitations to the expressibility of the ansatz hence higher error.

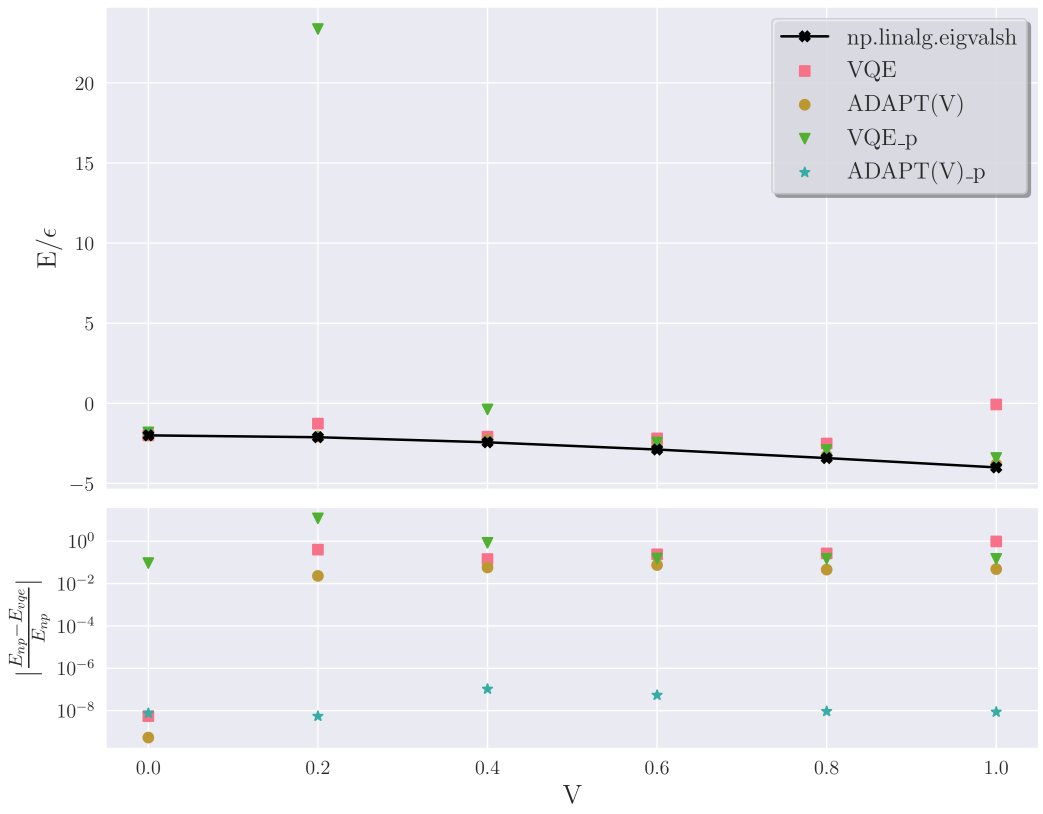


Figure 6.5: The LMG model with $J = 2$ with **exact energy simulation**, showing comparison between VQE with 1 rep and ADAPT-VQE with the V pool and maximum 12 ADAPT iterations. Both optimised with `BFGS` method.

6.5.4 Optimisation Methods Comparison

In this subsection we will delve into an analysis for the optimisers used for the VQE and ADAPT-VQE. All optimisers were capped at 10000 maximum iterations, which is usually never reached. We will choose the best optimisers for both the fixed-form ansatz and the ADAPT-VQE for the main results depending on whether shot noise is present or not.

Best Optimisers for Fixed-Form Ansatz

This is tested using the qubit Hamiltonian from Equation (5.14) and (5.16) for both the $J = 1$ and $J = 2$ cases, which was mapped to a 1 and 2 qubits Hamiltonian respectively.

Exact Energy Simulation Figure ?? shows the comparison between different optimisers for the fixed-form ansatz with `Rx` and `Ry` gates without shot noise.

The $J = 1$ case suggests that Powell is able to achieve the lowest error and $J = 2$ case suggests the same, for where the ansatz is expressive enough. Therefore the `Powell` method will be the

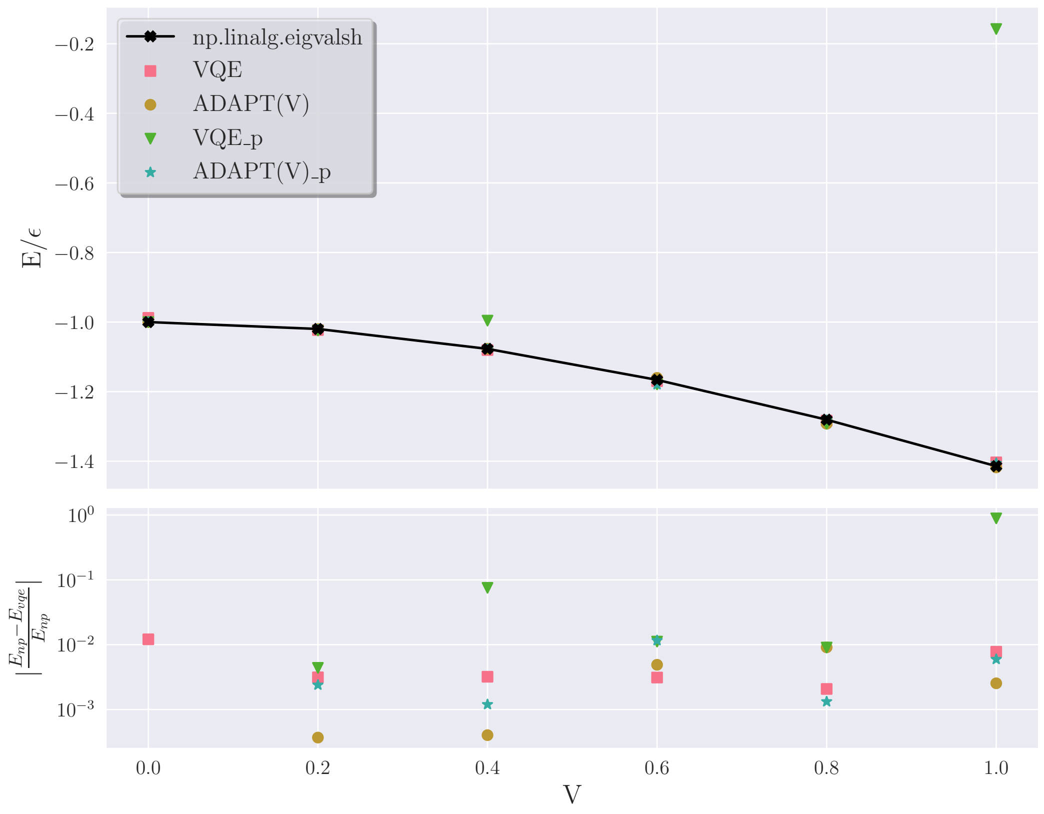


Figure 6.6: The LMG model with $J = 1$ with **ideal simulation**, showing comparison between VQE with 1 rep and ADAPT-VQE with the V pool and G pool and maximum 12 ADAPT iterations. The ADAPT-VQE was optimised with the `COBYLA` method and the normal VQE was with `Powell`.

choice of optimiser for the fixed-form ansatz for the main exact energy results.

Ideal Simulation Figure 6.9 shows the comparison between different optimisers for the fixed-form ansatz with shot noise. The most robust optimisers for the fixed-form ansatz with shot noise are `COBYLA` and `Powell`. The `Nelder-Mead` method produced reasonable results for some interaction strength but struggles at large interaction strength. Since Powell has better performance for both cases, it will be the choice of optimiser for the main results. p

Best Optimisers for the qubit-ADAPT-VQE

Exact Energy Simulation Figure 6.10 shows the comparison between different optimisers for the ADAPT-VQE without shot noise. We could see that BFGS and L-BFGS-B were able to achieve the lowest energy errors.

As seen in Figure 6.11, all the optimisers were able to achieve a high accuracy with just one iteration for the $J = 1$ case, and for the $J = 2$ case it took only 3 iterations for all ADAPT to converge regardless of the choice of optimisers.

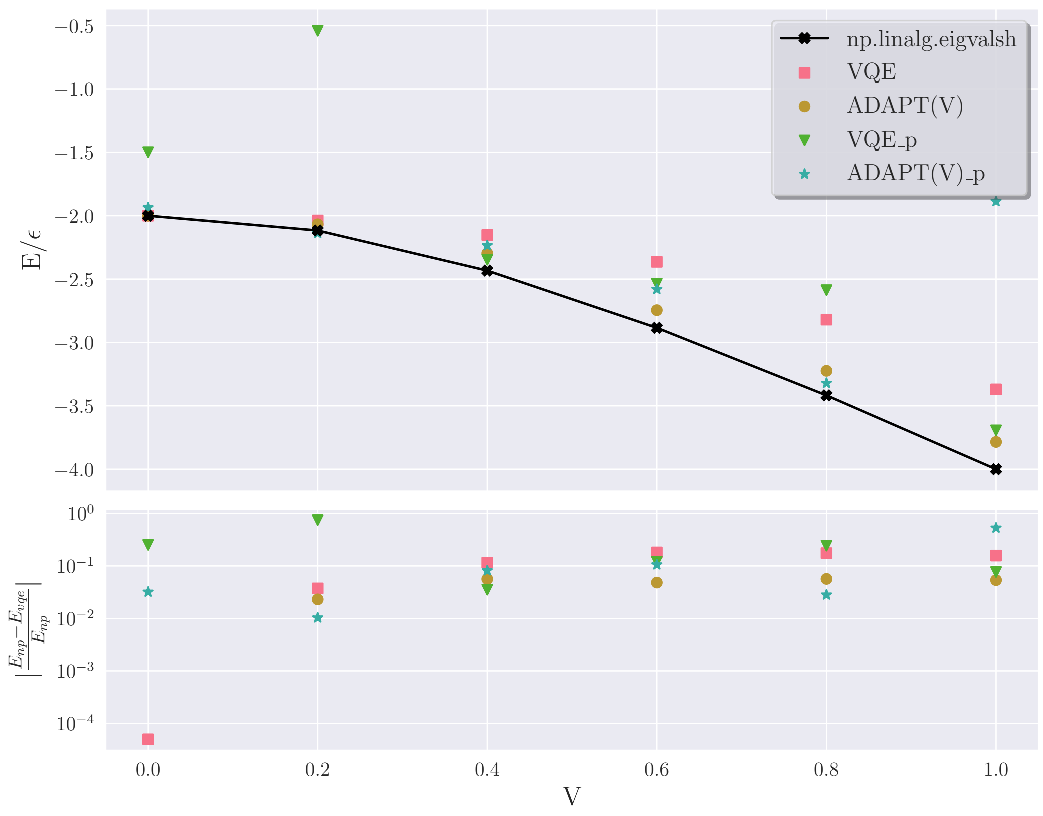


Figure 6.7: The LMG model with $J = 2$ with **ideal simulation**, showing comparison between VQE with 1 rep and ADAPT-VQE with the G pool and V pool and maximum 12 ADAPT iterations. The ADAPT-VQE was optimised with the `COBYLA` method and the normal VQE was with `Powell`.

ideal Simulation As it turned out, the best optimisers that is robust against shot noise is different from the one for without shot noise. Whilst we see the all optimisation method converges to different accuracies for the case without shot noise, some optimisers have shown to be more robust against shot noise. Noticeably, Nelder-Mead takes much longer to converge compared to other methods.

Even for $N = 10^4$ shots, most of the optimisers struggles to converge to the correct values. Since only `COBYLA` and `Powell` are able to converge to the correct values, they will be the choice of optimisers for the ADAPT-VQE for the main results. Interestingly, they seem to converge to the same values as shown in Figure 6.12.

To rule out the possibility that the optimisers which failed to converge were stuck in a local minimum, we could plot the energy landscape for the $J = 1$ case since there is only one qubit. The energy landscape is shown in Figure 6.13. In this case we only need Rx(θ) and Ry(ϕ) to span the entire Hilbert space, and the state is then $R_x(\theta)R_y(\phi)\,|0\rangle$. The energy landscape shows that there is only a single minimum (within a period), which is the one we are interested in. It is simply a result of the shot noise that the optimisers do not converge properly, which agrees with Figure 6.10, as all the optimisers were able to converge after 1 iteration.

Table 6.1: Table comparison between the fixed-form ansatz and the ADAPT-VQE for both Hamiltonians for the $J = 2$ case with $V = 1$.

	Parameters	**Gates**	**CNOT gates**	**Energy Function Evaluation**
Hardware Efficient	8	11	3	1654
qubit-ADAPT	12	128	46	846
Hardware Efficient (Pauli)	6	8	2	549
qubit-ADAPT (Pauli)	12	66	14	801

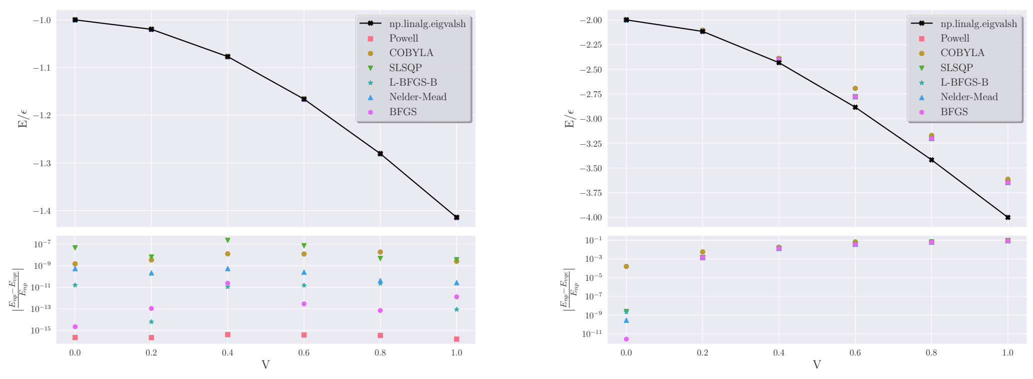


Figure 6.8: Comparison amongst different optimisers for the fixed-form ansatz with **exact energy simulation** for $J = 1$ and 2 .

6.6 Pairing Model

In this section we show results of the Pairing model with 4 doubly degenerated states and 4 particles for both the exact energy and ideal simulation. There are 18 terms in the qubit Hamiltonian for the pairing model. Therefore the expected error for the pairing model is 0.013. We have chosen $\xi = 1$ and the energy plotted is normalised by ξ and is hence unitless.

We first look at the results for $g \in [-1.5, 1.5]$ for the ADAPT-VQE with both the V and G pool, as well as the hardware efficient ansatz. As we can see from Figure 6.14, when exact energy is calculate, the ADAPT-VQEs with both pools were able to converge to the exact energy diagonalisation n with errors of orders of 10^{-9} for $g \in [-1.5, 1, 5]$ for only 8 maximum iterations. The G pool perform slightly worse (with error $\sim \mathcal{O}(10^{-6})$) for values close to 0. The hardware efficient ansatz was not able to obtain the same level of accuracies as the ADAPT-VQE, but the relative error was consistent for most of the values of g in orders of 10^{-2}, with the exception of $g = 0$ where the Hamiltonian given by Equation (5.27) is diagonal, which the hardware efficient ansatz was able to represent exactly.

We ran the exact energy simulation again for $g \in [-20, 20]$, as shown in Figure 6.15. The number of maximum iteration was set to 6 to match the parameter in the hardware efficient ansatz with 2 reps. For values of $g \in [-20, 0]$,the ADAPT-VQEs performed similarly to the VQE, with error between 10^{-2} to 10^{-1}. However, for large pairing strength g, the ADAPT-VQEs perform much much better than the VQE. The reason why the ADAPT-VQEs do not perform as well for the negative region of g could be due to the fact that 6 maximum iterations were not enough for the ADAPT-VQE to converge. Another point worth noting is that the bottom plot in Figure **??** is the relative error. As the energy increases, the energy increases in

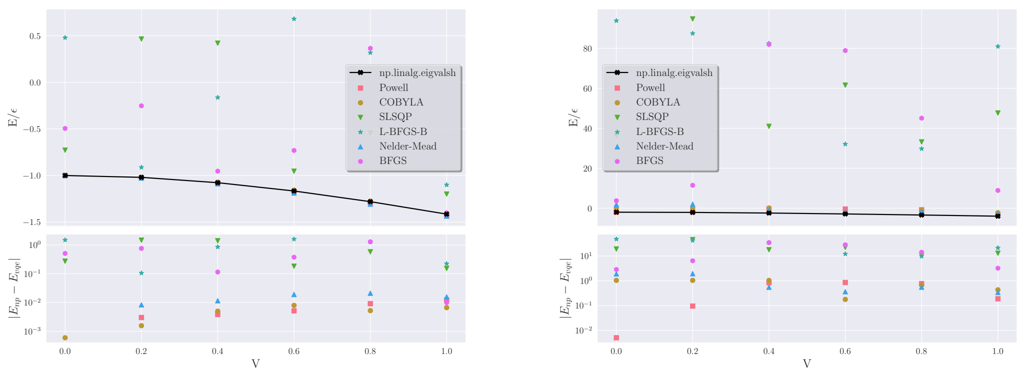


Figure 6.9: Comparison amongst different optimisers for the fixed-form ansatz with 100000 shots for $J = 1$ (left) $J = 2$ (right).

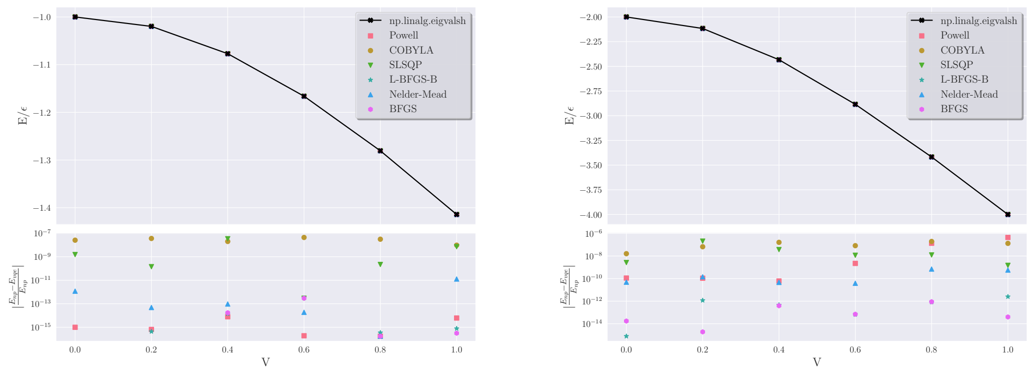


Figure 6.10: Comparison amongst different optimisers for the ADAPT-VQE without shot noise for $J = 1$ (top) and $J = 2$ (bottom).

magnitude, and the relative error takes account into the fact of the changing magnitude of the energy, making it a better indicator than the absolute error for algorithmic performance.

Figure 6.16 shows the results for the ideal simulation of the pairing model. We expected that by including shot noise the ADAPT-VQEs would require more iterations to converge, hence the maximum iteration was increased to 16.

For $g \in [-1.5, 1.5]$, the results produced with both ADAPT-VQEs and the VQE performed similarly in terms of accuracies. However, the ADAPT-VQEs were more consistent in convergence to the ground state while for many points the VQE did not manage to converge. This is aligned with the analysis we performed earlier for the LMG model for Figures 6.5 and 6.7 where for some points the VQE with hardware efficient ansatz failed not converge.

Results for ideal simulation for $g \in [0, 20]$ is presented in Figure 6.17. This is the only time we observed a difference in the performance of the pools. We observed similar results in Figure 6.15 but fewer iterations were allowed and the differences were not as clear. The G pool consistently outperformed the V pool in the interval $g \in [0, 20]$, with errors within orders of 10^{-2}, while the V pool performed similarly to the VQE with hardware efficient ansatz. Looking into the details we included 2 adapt iteration graphs shown in Figure 6.18 for $g = 8$ and $g = 16$ respectively. In both case, the G pool find the operators which allow it to converge after just a couple iterations,

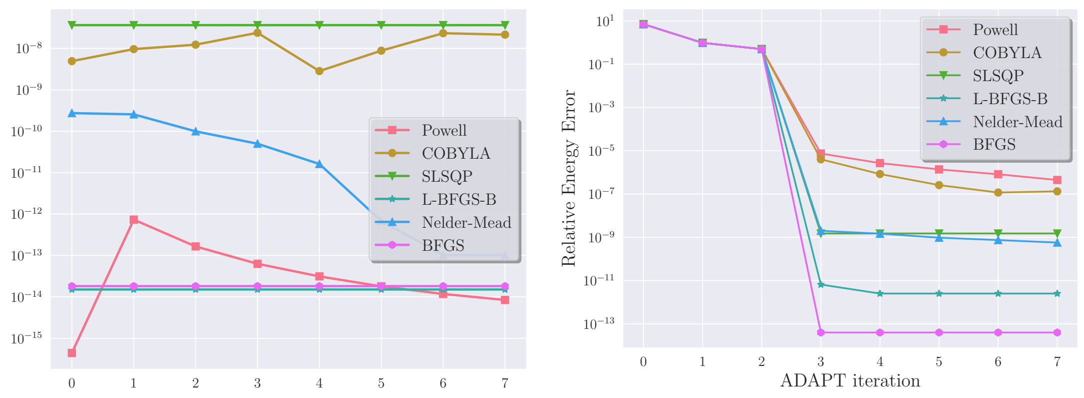


Figure 6.11: Comparison amongst different optimisers for the ADAPT-VQE without shot noise for $J = 1$ and $J = 2$.

Table 6.2: Optimisersation Results for $J = 1$ and $v = 1$

	Function Calls	**Run Time (s)**	**Relative Error**
`COBYLA`	399	2.481	0.00268
`Powell`	2475	10.849	0.01498
`SQSLP`	172	0.871	0.3144
`BFGS`	1160	3.749	0.286
`L-BFGS-B`	1145	4.190	0.301
`Nelder-Mead`	219625	749.491	0.263

whereas for the V pool, appending new operators did not improve the energy. For the $g = 8$ case, the V pool did not converge at all and for the $g = 16$ case, the energy stars to decrease at around 11 iterations, much later than the G pool. Looking at the structure of the pools as presented in equation (**??**), we could see that the operators in the G pool has a localised structure but it is unclear what is the cause of the difference in performance.

6.7 Deuteron Model

In this section we present both the exact energy simulation and ideal simulation results for the deuteron model for different number of basis dimension N. We compare our results to the numerical diagonalisation using *numpy.linalg.eigvalsh*. For the deuteron model with $N = n$, the number of qubits required is $\lceil \log_2(n) \rceil$. Unlike the other models we have simulated, the deuteron model is The qubit Hamiltonians will have different number of terms for different values of N, and the expected error depends on N as well. This is an interesting case to include as we can easily compare the performance for different number of qubits for the same system. The numbers of terms in the Hamiltonian are $8, 20$ and 48 for the $2, 3$ and 4 qubit cases respectively, so the expected error for these cases should be $\mathcal{O}(10^{-2})$.

6.7.1 Exact Energy Simulation

We first look at the results for $N \in [2, 32]$ for the exact energy simulation.

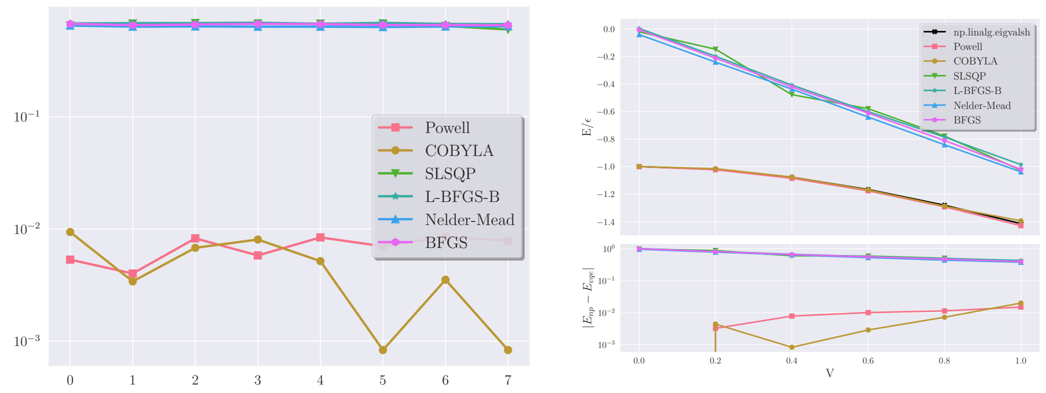


Figure 6.12: (Left) Comparison amongst different optimisers for the ADAPT-VQE, relative error per ADAPT iterations for 8 iterations with $10e_4$ shots for $J = 1$ at $V = 0.4$. (Right) Energy plot for different optimisers for different values of V.

Again, we saw that the errors obtained using the VQE with hardware efficient ansatz stay around 10^{-2}, even for more qubits. While the ADAPT-VQEs were outperforming the VQE by a large margin for cases with few qubits, as the number of qubits grow, this margin reduces drastically. For $N = 32$, the ADAPT-VQEs both have errors in orders of 10^{-4}. This is likely due to the fact that as the number of qubits grow, the number of operators needed to represent the ground state also grow. Since the minimum size of complete pool grows linearly with the number of qubits, the number of operators in the ansatz to represent the ground state should grow linearly as well. We will investigate this further in the Subsection 6.7.3.

6.7.2 Ideal Simulation

6.7.3 Scaling of Iterations with Number of Qubits

How many ADAPT iterations in the exact energy simulation do we expect the qubit-ADAPT-VQE to need to converge? We ran the exact energy simulation for the deuteron model for 1 to 7 qubits, using the `SLSQP` optimiser. The results are shown in Figure 6.21. Combining this with Figure 6.11 we could see the number of iterations required for the ADAPT-VQE to converge is irrelevant of the optimisers used but rather the number of the qubits in the system as shown in Figure 6.21. We summarised the results in Table **??**. Interestingly, the number of iterations grows much faster for lower number of qubits, and as the number of qubits increases, the number of iterations required to converge increases at a much slower rate, seemingly linearly. Due to the large amount of time required to run this simulation, we will not be able to run the convergence test for more qubits. However, if the number of iterations required for the convergence does not grow as fast as the number of qubits, then both the QubitAdaptAnsatz and the ADAPT-VQE will scale nicely into larger systems.

Finally, in an attempt to improve results further, we employed a new initialisation method, where the optimised state from the VQE is used as an initial state of the ADAPT-VQE. This is possible on actual hardware as well, since we knew the structure of the ansatz and the optimal parameters. One could simply run the hardware efficient ansatz circuit with the optimal parameters before starting the ADAPT iterations.

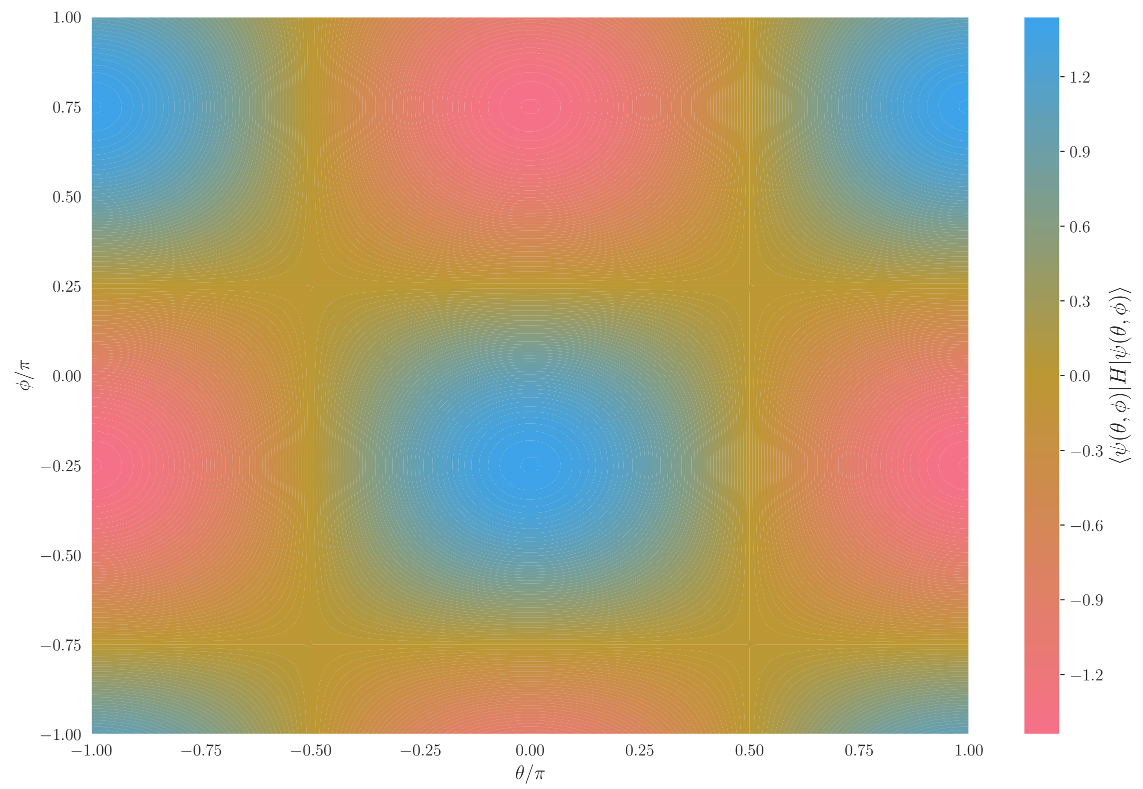


Figure 6.13: Energy landscape for $J = 1$ with $V = 1$.

Table 6.3: Number of iterations required for the ADAPT-VQE to converge for different number of qubits.

N	Number of Qubits	Number of Iterations
2	1	1
4	2	2
8	3	6
16	4	8
32	5	9
64	6	11
128	7	12

6.7.4 Initial States

The initial state has a significant impact on the convergence of the ADAPT-VQE and VQEs in general. We have found that by utilising the HardwareEfficientAnsatz optimised state as the initial state, the ADAPT-VQE is able to converge to the correct state much faster, as shown in Figure 6.22. When a small number of maximum iteration (5) is used, the ADAPT did not converge to the ground state for more than 2 qubits due to the limitation on iteration number. However, when the state was initialised with the hardware optimised state, the ADAPT-VQE with both pools converged to the minimum with error to orders of 10^{-2} or lower, even when the Ry state did not converge to the minimum. This could be extremely useful as it is usually not expensive in terms both quantum and classical resources. This is similar to initialising it to the HF state, except this way the we could potentially start in an entangled state which the HF state often is not. We also do not rely on classical many-body methods, which should be preferable.

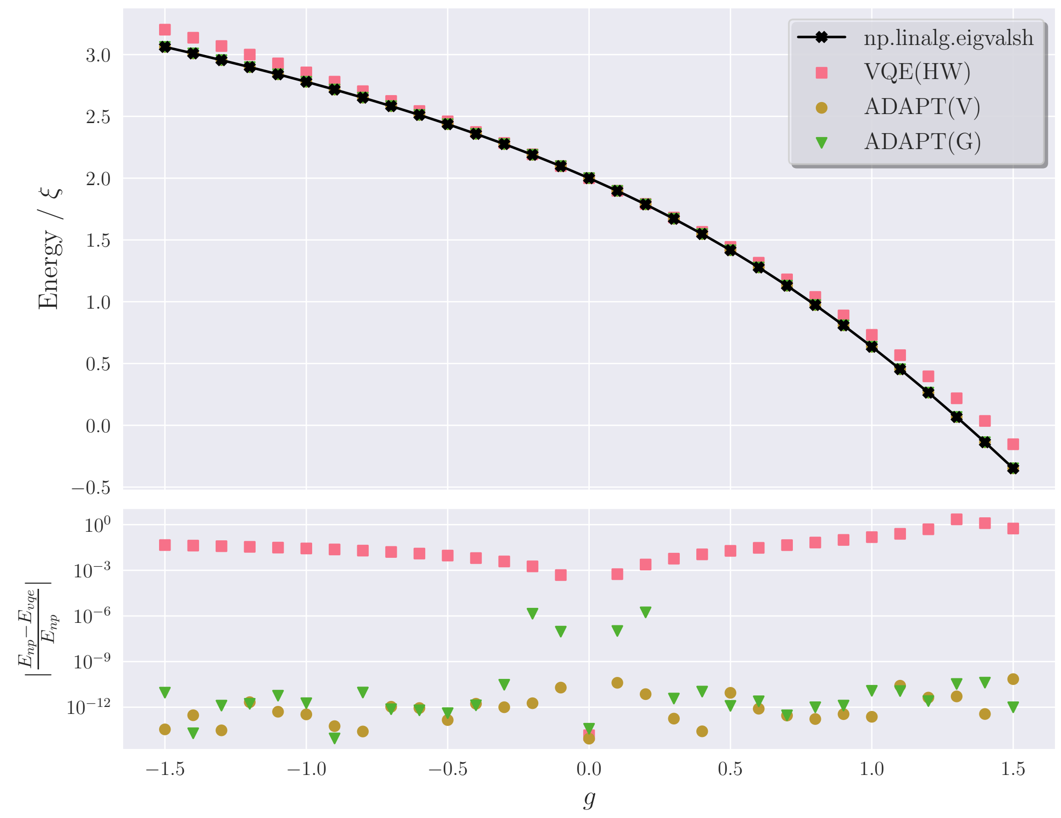


Figure 6.14: **Exact energy simulation** for the pairing model with `BFGS` optimiser for both the ADAPT-VQE and the VQE for a maximum of 8 iterations and $g \in [-1.5, 1.5]$. The hardware efficient ansatz with 2 reps was used for the VQE.

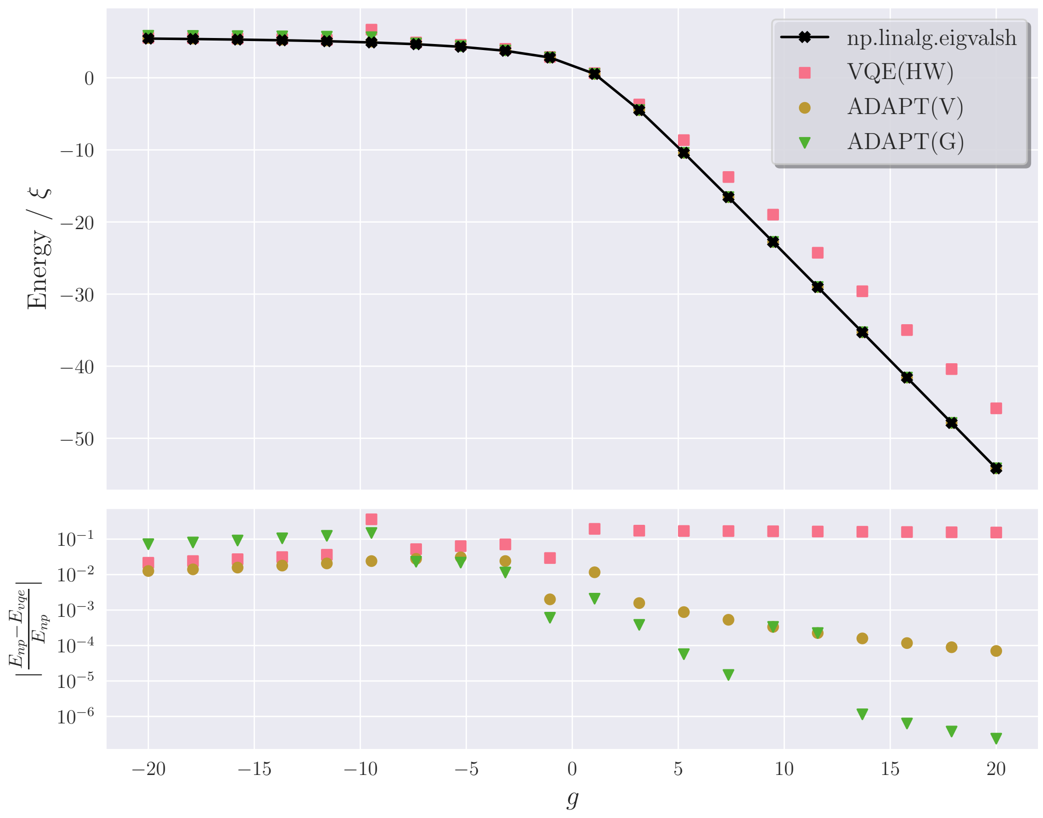


Figure 6.15: **Exact energy simulation** for the pairing model with BFGS optimiser for both the ADAPT and the normal VQE for a maximum of 6 iterations and $g \in [-20, 20]$. The HW ansatz with 2 reps was used for the VQE.

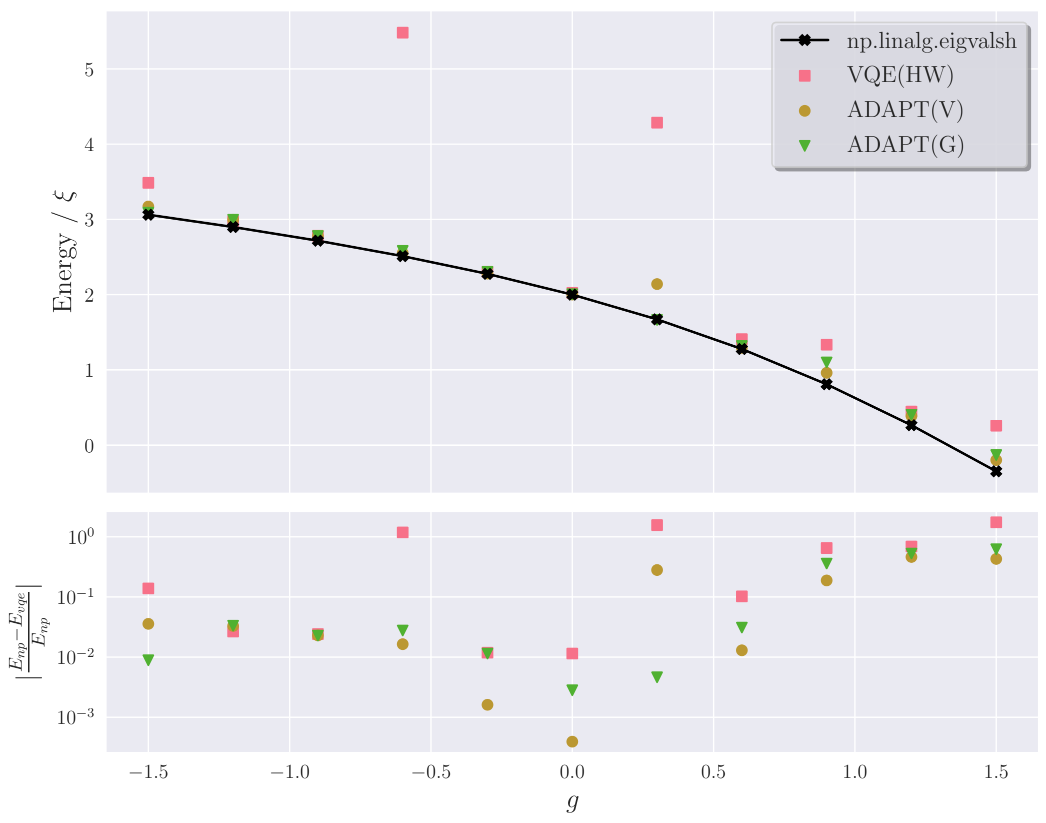


Figure 6.16: **Ideal Simulation** for the pairing model with `COBYLA` method for the ADAPT and `Powell` for the VQE for a maximum of 16 iterations for $g \in [-1.5, 1.5]$. The HW ansatz with 2 reps is used for the VQE.

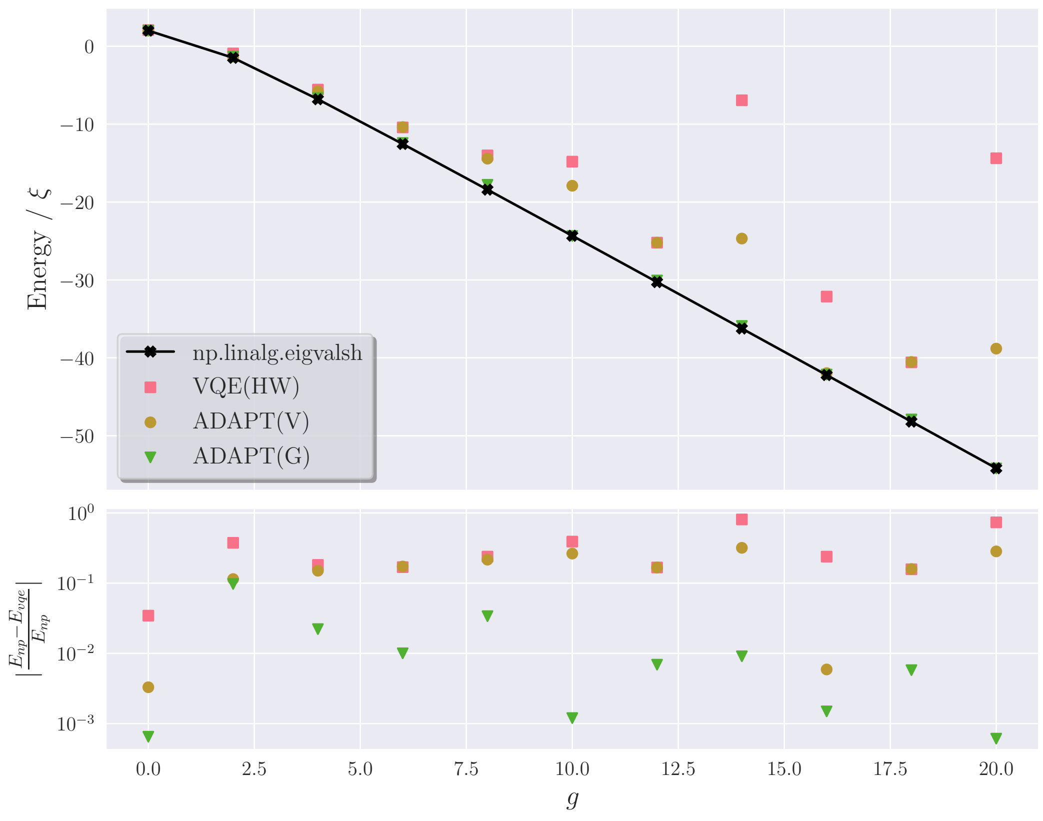


Figure 6.17: **Ideal Simulation** for the pairing model with `COBYLA` method for the ADAPT and `Powell` for the VQE for a maximum of 16 iterations for $g \in [0, 20]$

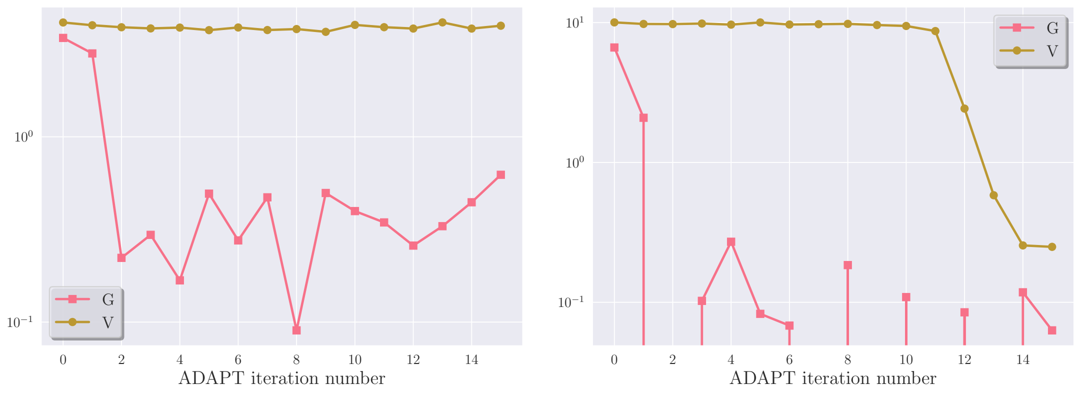


Figure 6.18: ADAPT-VQE error at every iteration for the pairing model using `COBYLA` optimiser for $g = 16$ (left) and $g = 18$ (right).

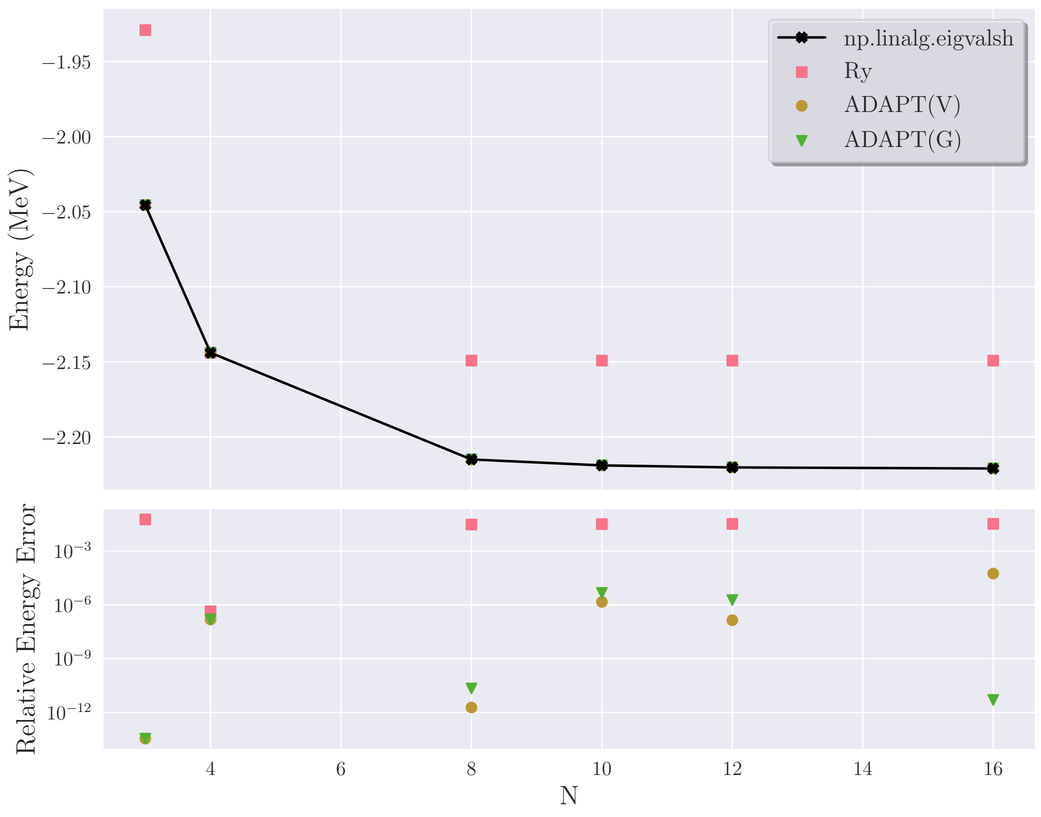


Figure 6.19: The deuteron model with **exact energy simulation** for $N \in [3, 16]$ with the `BFGS` optimiser for the ADAPT-VQE for a maximum of 16 iterations.

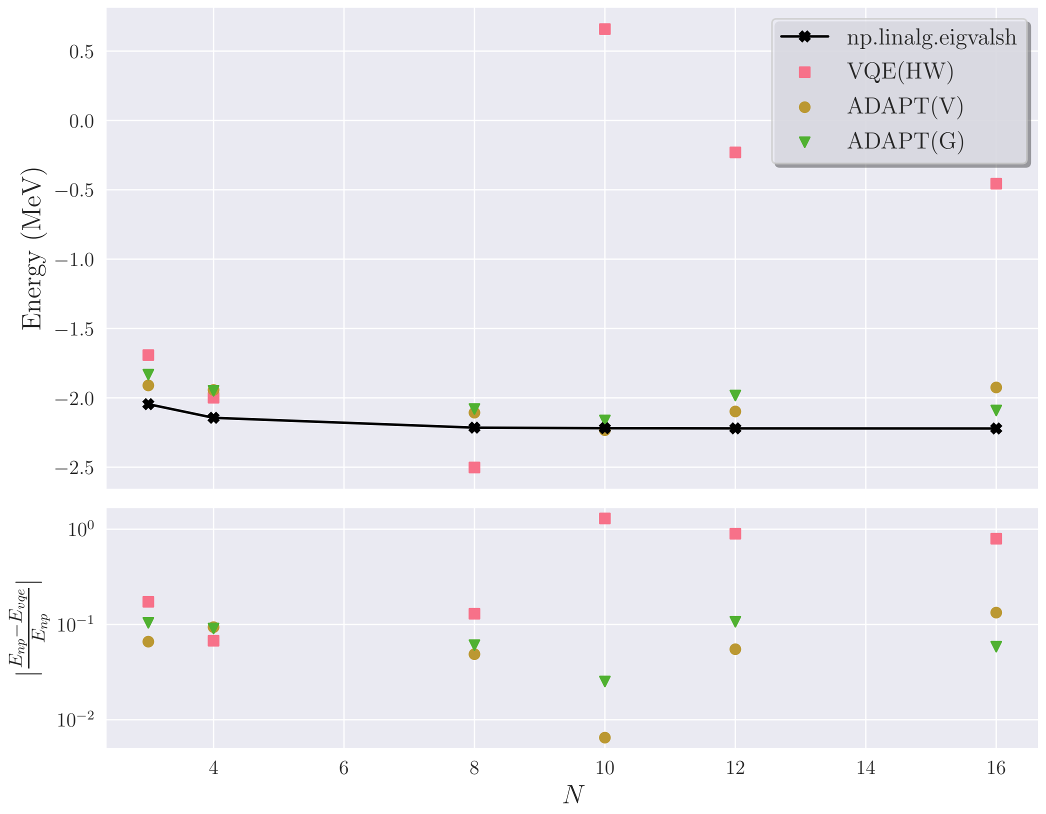


Figure 6.20: The deuteron model with **ideal simulation** for $N \in [3, 16]$ with the `COBYLA` optimiser for the ADAPT-VQE for a maximum of 16 iterations.

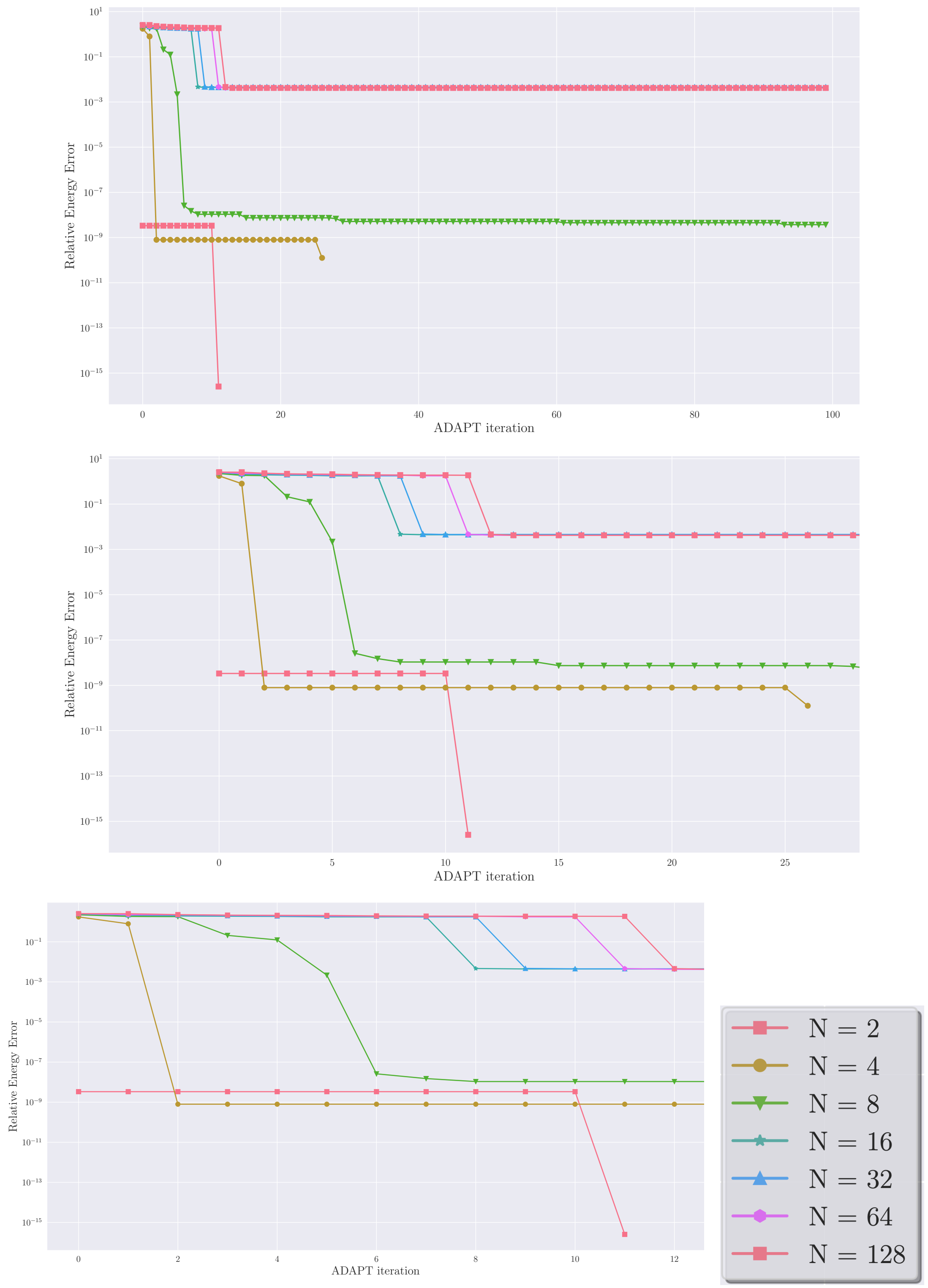
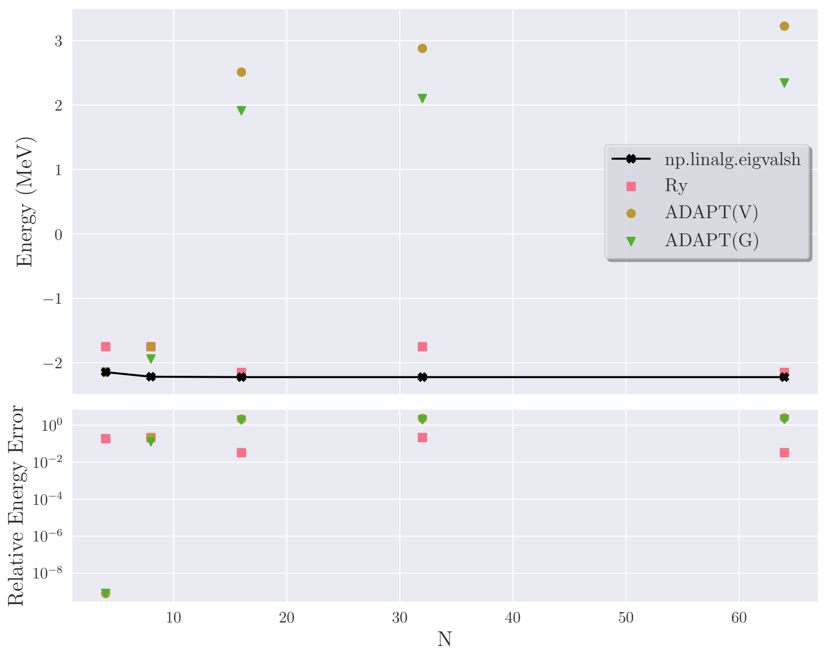


Figure 6.21: Number of ADAPT iterations for the deuteron model with exact energy calculation optimised with `SLSQP` for 1 to 7 qubits with maximum 100 iterations using the G pool. The pink line to the most left with lower errors correspond to the 1 qubit case, and the other pink like correspond to $N = 128$, the 7-qubit case. The top figure shows the whole 100 iteration, the middle figure shows a zoomed in version of the top figure for around 27 iterations, and the bottom figure shows a zoomed in version of the middle figure for around 13 iterations.



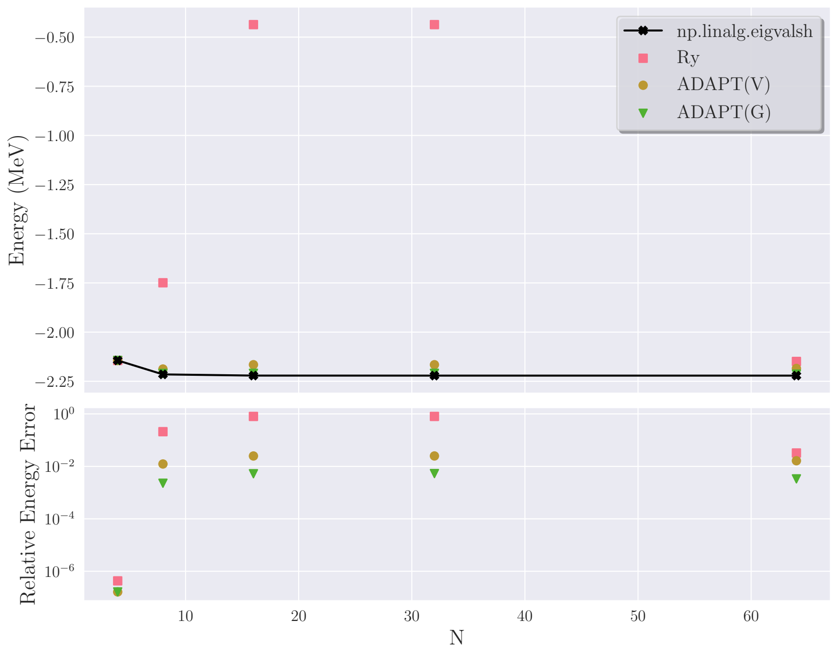


Figure 6.22: Results obtained for the deuteron Hamiltonian with different basis dimension N with **exact energy calculation** for 5 maximum iterations and 2 rep for the Hardware Efficient Ansatz Ry. The exponentials were decomposed using the **staircase** algorithm, and both VQEs were optimised with the `SLSQP` method. The top figure shows the results when the initial state is the maximally superposed state , and the bottom figure shows the results when the initial state is the Hardware Efficient Ansatz optimised state.

Chapter 7

Conclusion

7.1 Summary of Results

The aim of this thesis project is two fold: to create a quantum computing library that is oriented towards physicist and to study the performance of the Variational Quantum Eigensolver (VQE) and the Adaptive, Problem Tailor (ADAPT) VQE to obtain some insights for the best practices in using these algorithms.

We built Quanthon as a result, which contains the basic elements to be able to perform any quantum operation in principal. The `Hamiltonian` class was made to allow the Hamiltonians to be entered in a low level way by specifying the one- and two-body coefficients or in matrix representation. Other functionalities include the `VQE` and `ADAPT-VQE` classes which are the main focus of this project, as well as different modules that can assist the quantum simulations, including the mappers to convert fermionic Hamiltonians or matrix Hamiltonians to qubit Hamiltonians, the staircase and inverted staircase algorithms for converting exponentials of Pauli strings to quantum circuits, and the expectation value estimator which simulates the measurements of the quantum circuits and computes the expectation values of the Hamiltonians.

With the library built, we then proceeded to study the performance of the VQE with hardware efficient ansatz and ADAPT-VQE with two different minimal complete pool, the V and the G pool, through both exact energy and ideal simulations of the multiple models. With the exact energy simulation, we observed that the hardware efficient ansatz in many cases does not converge to the ground state. When it does, the relative error is usually of order 10^{-2}. The ADAPT-VQE with minimal pools, however, can indeed converge to the ground state with relative error lower than 10^{-6} given enough ADAPT iterations. The number of iterations it takes for the qubit-ADAPT-VQE to converge scales roughly linearly with the number of qubits in the model and the choice of the optimiser is irrelevant for the number of iterations required. The best optimiser for the VQE was found to be the Powell method, and the best optimiser for the qubit-ADAPT-VQE was the BFGS method.

In the presence of shot noise, the convergence of the qubit-ADAPT-VQE slows down, but is still more stable and consistently outperforms the hardware efficient ansatz. We compared the error with the expected error from the number of shots to see if the algorithms have converged. The best optimiser in this case is the COBYLA method for the qubit-ADAPT-VQE and the Powell method for the VQE. We noticed that the ADAPT-VQE almost never exits due to the gradient being below the tolerance level, which is likely due to the noise.

No significant differences were observed in the performances of the V and G pool, except in the case of large pairing strength the G pool converges much faster than the V pool.

Additionally, we found that the gradient the entire minimal pool operators can be 0 for many Hamiltonians if the state is initialised in the $|0\rangle$ state. This cause the ADAPT-VQE to exit immediately without any iterations. Initialising randomly avoids this problem but causes slow convergence. First we used the maximally superposed state as an initial state. Later, we found that by initialising the ADAPT-VQE with the optimised state from hardware efficient ansatz boosts performances by reducing the number of iterations it takes for the ADAPT-VQE to converge.

7.2 Future Work

We will group the future work into two categories: improvements to the Quanthon library and improvements to the VQE and ADAPT-VQE algorithms.

7.2.1 Improvements to Quanthon

Many functionality can be added to the Quanthon Library, such as adding more predefined gates or other popular algorithms. We could also improve the speed to simulation great by parallelising the measurements and a function which groups commuting terms of the Hamiltonian to be measured simultaneously. Integration with quantum hardware could also be extremely useful to allow the library to run on real quantum devices. More ansatz could be implemented to allow for flexibility and performance comparison, specifically the unitary coupled cluster ansatz [**Romero2019**] and the fermionic-ADAPT ansatz [**grimsley2019**]. This would allow for a more comprehensive comparison of the ansatz. Another interesting direction would be to implement noise model to allow for noisy simulations.

7.2.2 Improvements to Simulation Results

The ideal simulation takes a long time to run, therefore the maximum number of iterations were all set to below 30, the performance of the ADAPT-VQE could potentially be improved by simply allowing the algorithm to run for more iterations. The 0 gradient problem is also concerning as the initial state could cause the algorithm to exit prematurely. More research is needed to find out under which circumstance would they gradient vanish and come up with either a different operator selection criteria for choosing an operator to be appended to the ansatz, or a systematic way to selecting the initial state to avoid the difficulty in convergence.

Appendix A

Properties of Pauli Matrices

The Pauli matrices are essential building blocks in quantum mechanics and quantum computing. They are defined as Equation (**??**)

A.1 Commutation Relations

The commutation relation for the Pauli matrices σ_x, σ_y and σ_z is given by

$$[\sigma_i, \sigma_j] = 2i\epsilon_{ijk}\sigma_k, \tag{A.1}$$

where ϵ_{ijk} is the Levi-Civita symbol.

The anticommutation relation for the **fermionic** creation and annihilation operators a_i and $a_i^\dagger$ is given by

$$\{a_i, a_j\} = \left\{a_i^\dagger, a_j^\dagger\right\} = 0, \quad \left\{a_i, a_j^\dagger\right\} = \delta_{ij}. \tag{A.2}$$

A.2 Pauli Matrices as Basis

The Pauli matrices σ_x, σ_y and σ_z form a basis for the space of 2×2 matrices. Any matrix M,

$$M = \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix}.$$

can be written as a linear combination of the Pauli matrices,

$$M = (a\mathbb{I} + b\sigma_x + c\sigma_y + d\sigma_z),$$

$$a\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + b\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} + c\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} + d\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

the coefficients a, b, c and d can be found by solving the linear equations:

$$\alpha = a + d, \tag{A.3}$$
$$\beta = b - ic, \tag{A.4}$$
$$\gamma = b + ic, \tag{A.5}$$
$$\delta = a - d. \tag{A.6}$$

Solving Equations (A.3) for a, b, c and d gives

$$a = \frac{1}{2}(\alpha + \delta) = \frac{1}{2}\mathrm{Tr}(IM), \tag{A.7}$$
$$b = \frac{1}{2}(\beta + \gamma) = \frac{1}{2}\mathrm{Tr}(XM), \tag{A.8}$$
$$c = \frac{1}{2i}(\gamma - \beta) = \frac{1}{2}\mathrm{Tr}(YM), \tag{A.9}$$
$$d = \frac{1}{2}(\alpha - \delta) = \frac{1}{2}\mathrm{Tr}(ZM). \tag{A.10}$$
$$\tag{A.11}$$

In fact, this generalises. For a new matrix A of size $2^n \times 2^n$, given that $\{P_i\}$ is a set of orthonormal basis of the dimension. Then A can be written as a linear combination of the basis $\{P_i\}$,

$$A = \sum_j a_j P_j.$$

Products of Pauli matrices is given in Table A.1. It is not hard to see that the trace of these products can be expressed as

$$\mathrm{Tr}(\sigma_j \sigma_k) = 2^n \delta_{jk},$$

since the Pauli matrices are traceless. The 2^n term comes from the trace of the identity matrix in 2^n dimension. For an arbitrary coefficient a_k corresponding to the basis B_k, following the above equation we have

$$\begin{aligned}
P_k^\dagger A &= P_k^\dagger \sum_j a_j P_j \\
P_k^\dagger A &= \sum_j a_j \mathrm{Tr}\left(P_k^\dagger P_j\right) && \text{Linearity of summation} \\
\mathrm{Tr}\left(P_k^\dagger A\right) &= \sum_j a_j \mathrm{Tr}\left(P_k^\dagger P_j\right) && \text{Linearity of trace} \\
\mathrm{Tr}\left(P_k^\dagger A\right) &= \sum_j 2^n a_j \delta_{jk} \\
a_k &= \frac{1}{2^n}\mathrm{Tr}\left(P_k^\dagger A\right)
\end{aligned}$$

Tensor products of Pauli matrices form a basis for the space of $2^n \times 2^n$ matrices due to the linearity of the tensor product. This proves Equation (3.26).

A.3 Product of Pauli Matrices

Table A.1 shows the matrix products of pairs of Pauli matrices.

Table A.1: Products of two Pauli matrices (Row × column).

	σ_x	σ_y	σ_z
σ_x	I	$i\sigma_z$	$-i\sigma_y$
σ_y	$-i\sigma_z$	I	$i\sigma_x$
σ_z	$i\sigma_y$	$-i\sigma_x$	I

Appendix B

The Vanishing ADAPT gradient

In this appendix we will show that the operator gradient given by Equation (3.28) for the 3-qubit qubit-ADAPT pool is 0 for the $|0\rangle$ state using the example Hamiltonian given in Equation (6.4).

We showed in Section 6.3 that the gradient of the state $|0\dots0\rangle$ is 0 for one of the operators, $iYZZ$. We will show that this is true for the other three operators in the pool, given in Equation (B.1), (B.2), (B.3).

$$iIYZ = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \tag{B.1}$$

$$iIIY = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \end{pmatrix} \tag{B.2}$$

$$iIYI = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \end{pmatrix} \tag{B.3}$$

The following code snippet is the code used to numerically calculate the gradient for each operator with the same Hamiltonian,

$$\langle\psi|[H, A]|\psi\rangle$$

```python
from Quanthon import pauli_sum, QubitAdaptAnsatz
from Quanthon.utils import get_pauli

def comm(A,B):
    if type(A) == str:
        A = get_pauli(A)
    if type(B) == str:
        B = get_pauli(B)

    return A@B - B@A
def grad(A, h_mat, state):
    return state.conj().T @ comm(h_mat, get_pauli(A)) @ state

qh = [('IIZ', 2), ('IXX', -0.5), ('IYY', 0.5), ('ZZI', 2)]
H = pauli_sum(qh)

qa = QubitAdaptAnsatz(3)
pool = qa.pool

for A in pool:
    A = A.strip('i')
    print(grad(A, H, Qubits(3).state))
```

The output of the code is as follows:

```
0j
0j
0j
0j
```

This is a curious phenomenon as I have not seen it in relevant literature. To investigate this further, we will construct a 3-qubit Hamiltonian randomly while making sure every term possible exist. The code snippet is shown below. Since the pool is complete for all real states, we will also ignore terms with odd number of Y operators as they will not appear in a real Hamiltonian. The function `generate_hermitian_matrix` generate a random Hermitian matrix and `has_odd_y` is defined to g check if a given Pauli string has an odd number of Y operators.

```python
import numpy as np

n = 3
random_h = generate_hermitian_matrix(2**n)
all_ops = pauli_decomposition(random_h)
pool = QubitAdaptAnsatz(n).pool

# the zero state
state = np.zeros(2**n)
state[0] = 1

grad_zero_terms = []
```

```python
real_terms = len(all_ops)

for term, coeff in all_ops:
        h_mat = get_pauli(term)

        if has_odd_y(term):
                real_terms -= 1
                continue
        zeros_count = 0
        for A in pool:

            g = grad(A, h_mat, state)
            if g == 0:
                    zeros_count += 1
            if zeros_count == len(pool):
                    grad_zero_terms.append(term)

# print(grad_zero_terms)
print(f'{len(grad_zero_terms)} out of {real_terms} terms have zero
   gradient for all operators of the pool.')
```

The output of the code is as follows:

```
24 out of 36 terms have zero gradient for all operators of the pool.
```

and for $n = 4$

```
104 out of 136 terms have zero gradient for all operators of the pool.
```

This means initialising the state to $|0 \dots 0\rangle$ will result in a zero gradient for a significant number of terms in the Hamiltonian, causes the ADAPT-VQE to immidiately discard stop without converging to the actual ground state. We observed that by initialising the state randomly, while still keeping it real, the gradient is non-zero for all terms, except trivially the `I...I` term.

```python
state = np.random.rand(2**n)
```

However, during the simulation we observed that this initialisation can lead to a poor convergence of the ADAPT-VQE. Another initial state $|+^{\otimes n}\rangle$ has 58 out of the 136 terms with 0 gradient instead, but the convergence is much better than the random initialisation.

Appendix C

More Results

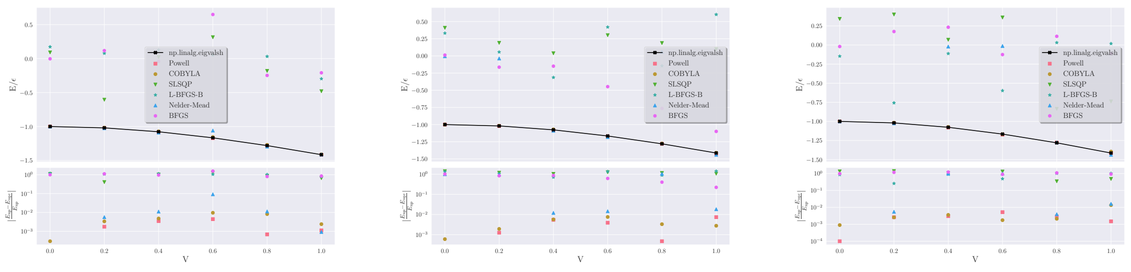


Figure C.1: Comparison amongst different optimisers for the fixed-form ansatz with **ieal simulation** for $J = 1$. Different number of shots were used across all three figures: (left) 1000, (middle) 10000 and (right) 100000.

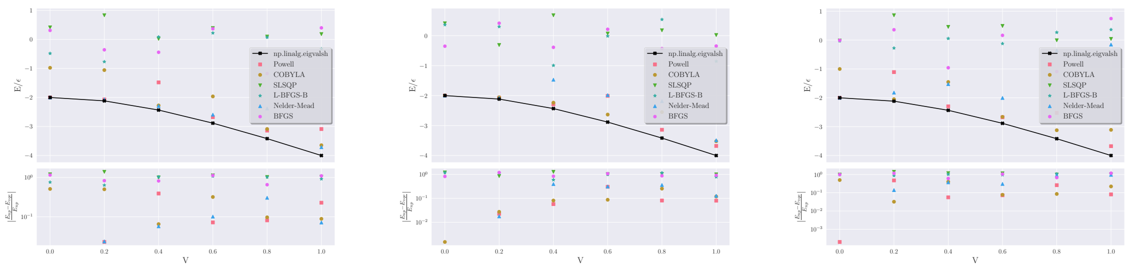


Figure C.2: Comparison amongst different optimisers for the fixed-form ansatz with **ideal simulation** for $J = 2$.

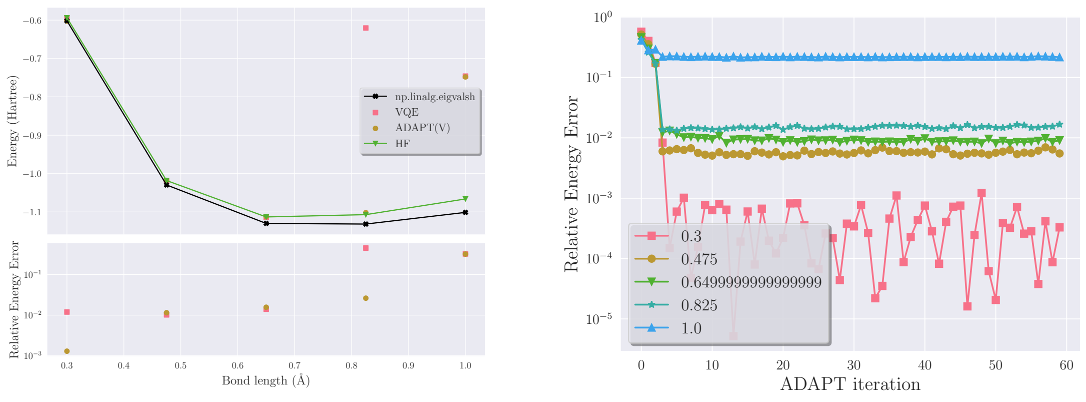


Figure C.3: (Left) The hydrogen molecule with **ideal simulation** with 100000 shots for a maximum of 60 ADAPT iterations. The ADAPT-VQEs were optimised with the `COBYLA` method and the VQE with the `Powell` method. The exponential was decomposed using the **inverted Staircase** algorithm. (Right) The error at every ADAPT iteration.

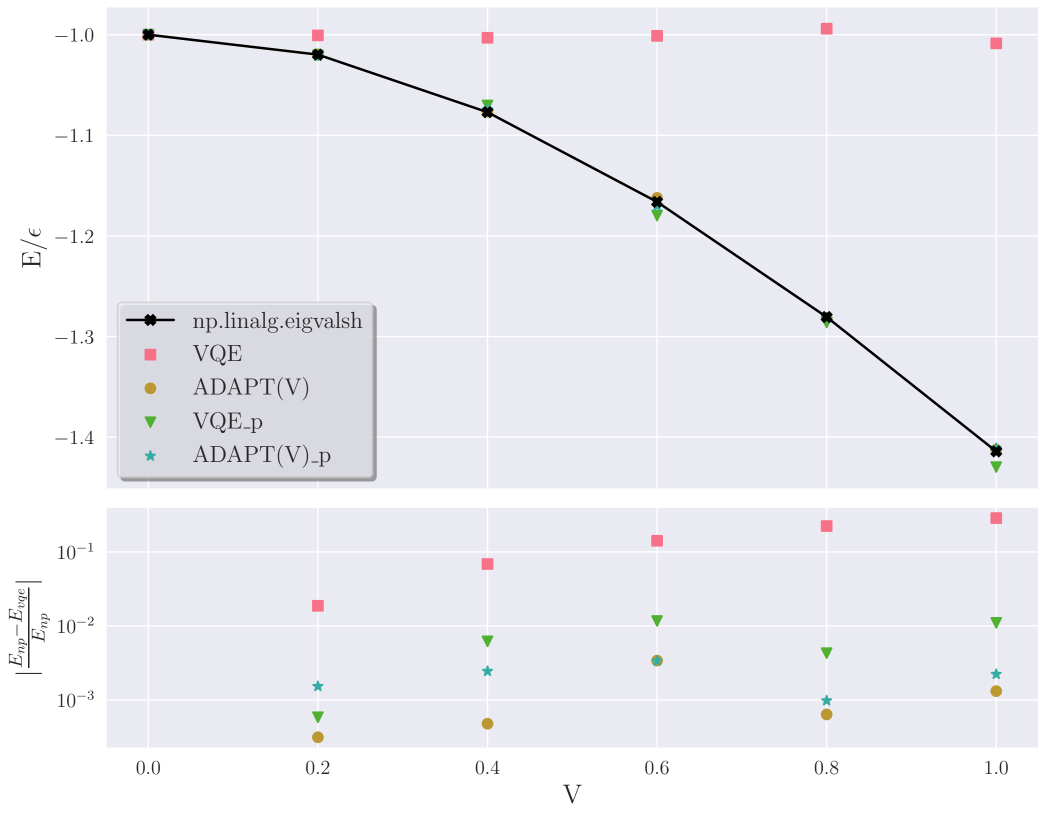


Figure C.4: The LMG model with $J = 1$ with **ideal simulation**, showing comparison between VQE with 2 rep and ADAPT-VQE with the V pool with the V pool and maximum 12 ADAPT iterations. The ADAPT-VQE was optimised with the `COBYLA` method and the normal VQE was with `Powell`.