This electronic thesis or dissertation has been downloaded from Explore Bristol Research, http://research-information.bristol.ac.uk

Author:
Mineh, Lana

Title:
Solving the Hubbard model using the variational quantum eigensolver

General rights
Access to the thesis is subject to the Creative Commons Attribution - NonCommercial-No Derivatives 4.0 International Public License. A copy of this may be found at https://creativecommons.org/licenses/by-nc-nd/4.0/legalcode This license sets out your rights and the restrictions that apply to your access to the thesis so it is important you read this before proceeding.

Take down policy
Some pages of this thesis may have been removed for copyright restrictions prior to having it been deposited in Explore Bristol Research. However, if you have discovered material within the thesis that you consider to be unlawful e.g. breaches of copyright (either yours or that of a third party) or any other law, including but not limited to those relating to patent, trademark, confidentiality, data protection, obscenity, defamation, libel, then please contact collections-metadata@bristol.ac.uk and include the following information in your message:

- Your contact details
- Bibliographic details for the item, including a URL
- An outline nature of the complaint

Your claim will be investigated and, where appropriate, the item in question will be removed from public view as soon as possible.

Solving the Hubbard model using the variational quantum eigensolver

By

Lana Mineh

A dissertation submitted to the University of Bristol in accordance with the requirements for award of the degree of Doctor of Philosophy (PhD) in the Faculty of Science, School of Mathematics

March 2022

Word count: Thirty-six thousand

Abstract

Simulating quantum-mechanical systems relating to quantum chemistry or solid-state physics is one of the most important problems that quantum computers are anticipated to tackle. However, near-term quantum computers will be noisy and have a limited number of qubits, restricting the types of computations that can be performed. This has lead to the development of hybrid quantum-classical algorithms, which are suitable for use on these near-term devices. Systematic studies involving high-performance classical simulations are required to thoroughly assess the effectiveness of these algorithms.

In this thesis, we address this challenge for the variational quantum eigensolver applied to the Hubbard model, an important model from condensed-matter physics used to describe the behaviour of correlated electrons. We consider both the solution of instances of this model directly and a compressed form obtained using density matrix embedding theory. All aspects of this hybrid quantum-classical algorithm are considered, from the initial encoding of the fermionic Hamiltonian onto the quantum computer, to the ansatz circuits and the classical optimisation routine.

For the purpose of speeding up future simulations of variational quantum algorithms, we develop a new high-performance quantum simulator. We describe the efficient algorithms used for performing gates and measurement. We also discuss how certain circuit properties, such as number or spin preservation, can lead to further algorithmic speed-ups. Implementing these techniques on high-performance classical hardware may facilitate the numerical exploration of larger problems.

Acknowledgements

I would like to extend a big thanks to my supervisor Ashley Montanaro for his support and guidance over the past three years – I always learned something new in our meetings.

Thanks to everyone at Phasecraft for making the PhD so interesting, with a special thanks to my co-authors Stasja Stanisic and Chris Cade. Thanks to all of the staff and students from the Quantum Engineering CDT, QETLabs and the quantum information theory group for making my time in Bristol so special. My particular thanks go to Oliver Thomas, Alex Qiu and Naomi Solomons for interesting discussions about science and programming; Friederike Jöhlinger and Huili Hou for their friendship; and the rest of my cohort – Andrés, Ankur, Ben, David, Dom, Frazer and Jake – the CDT wouldn't be the same without you all.

I would also like to thank my amazing partner John Scott for having the patience to proofread this entire thesis and for always being helpful. Finally, thanks to my parents Chro and Amin for their love and support.

Declaration

I declare that the work in this dissertation was carried out in accordance with the requirements of the University's *Regulations and Code of Practice for Research Degree Programmes* and that it has not been submitted for any other academic award. Except where indicated by specific reference in the text, the work is the candidate's own work. Work done in collaboration with, or with the assistance of, others, is indicated as such. Any views expressed in the dissertation are those of the author.

SIGNED: .... DATE: ....

Contents

Abstract . . . i
Acknowledgements . . . iii
Declaration . . . v

List of Figures **ix**

List of Tables **xi**

List of Abbreviations **xiii**

1 Introduction **1**
- 1.1 Quantum computing . . . 2
- 1.2 Noisy intermediate-scale quantum algorithms . . . 4
- 1.3 The Hubbard model . . . 7
- 1.4 Thesis outline . . . 10
  - 1.4.1 List of publications . . . 12

2 The variational quantum eigensolver **13**
- 2.1 Outline of the VQE algorithm . . . 13
- 2.2 Fermion-to-qubit encodings . . . 15
- 2.3 Ansatz circuits . . . 17
  - 2.3.1 The Hamiltonian variational ansatz . . . 19
  - 2.3.2 Fermionic swap networks . . . 22
- 2.4 Measuring the expectation value . . . 23
- 2.5 Classical optimiser . . . 26
  - 2.5.1 Simultaneous perturbation stochastic approximation . . . 28
  - 2.5.2 Coordinate descent algorithm . . . 29

3 Solving the Hubbard model using the variational quantum eigensolver **33**
- 3.1 VQE implementation details . . . 34
  - 3.1.1 Efficient ansatz circuit implementation . . . 38
  - 3.1.2 Measurement scheme . . . 41
  - 3.1.3 Implementation on realistic hardware . . . 43
- 3.2 Numerical results . . . 46
  - 3.2.1 Exact simulations . . . 48

3.2.2 Incorporating measurements . . . 54
3.2.3 Depolarising noise . . . 57
3.3 Summary . . . 58

4 Density matrix embedding theory applied to the Hubbard model 61
4.1 Introduction to DMET . . . 62
4.2 The single-shot embedding algorithm . . . 63
4.2.1 Calculating observables from the embedded Hamiltonian . . . 68
4.3 Form of the embedded Hamiltonian . . . 68
4.3.1 1D Hubbard model . . . 69
4.3.2 2D Hubbard model . . . 75

5 The variational quantum eigensolver as an embedded system solver 77
5.1 VQE implementation details . . . 78
5.1.1 Efficient ansatz circuit implementation . . . 78
5.1.2 Measurement scheme . . . 84
5.2 Numerical results . . . 87
5.2.1 Exact simulations . . . 88
5.2.2 Incorporating measurements . . . 93
5.3 Summary . . . 97

6 Simulating quantum computing 99
6.1 Storing the state vector . . . 101
6.2 Simulating one- and two-qubit gates . . . 103
6.2.1 One-qubit gates . . . 103
6.2.2 Two-qubit gates . . . 105
6.3 Simulating measurements . . . 107
6.4 Number-preserving simulator . . . 110
6.5 Summary . . . 113

7 Conclusion 115

Appendices 117

A The number-preserving ansatz 119

B Calculating the 1-RDM of a Slater determinant 123

References 125

List of Figures

1.1 The Hubbard model grid. 9

2.1 The hardware efficient ansatz circuit on n qubits. 18
2.2 Implementation of the hopping gate using two-qubit gates. 21
2.3 Generic swap network for four qubits. 23
2.4 Measurement of all hopping terms on six qubits. 25

3.1 Demonstration of the JW snake ordering on a physical architecture. 35
3.2 Four sets of commuting hopping terms in the Hubbard model. 36
3.3 Implementation of vertical hopping terms using fermionic swap networks. 40
3.4 Gates required to implement one layer of the EHV or NPr ansatz for a single spin-type. 41
3.5 Implementation of the EHV and NPr ansätze on the Google Sycamore architecture for even N_x. 44
3.6 Implementation of the EHV and NPr ansätze on the Google Sycamore architecture for odd N_x. 46
3.7 Ansatz depths required to represent the ground state of the Hubbard model for the HV, EHV and NPr ansätze. 50
3.8 Scaling of infidelity with number of layers of the EHV ansatz for grids with twelve sites. 50
3.9 Final fidelity achieved with varying U for the EHV ansatz with different grid sizes. 51
3.10 Depth of the EHV ansatz required to reach 0.99 fidelity with the ground state of the Hubbard model as a function of U. 52
3.11 Depth of the EHV ansatz required to reach 0.99 fidelity with the ground state of the half-filled Hubbard model. 53
3.12 Infidelity, absolute error with the actual ground state and absolute error with the true double occupancy for the EHV ansatz. 53
3.13 Comparison of standard and three-stage SPSA. 55
3.14 Infidelity reached during the optimisation process with CD and SPSA optimisers using realistic measurements. 56
3.15 Infidelity reached during the optimisation process with CD and SPSA optimisers for simulations using depolarising noise, with and without error detection. 58

4.1 Single-shot embedding compared with Bethe ansatz and exact diagonalisation of small instances of the Hubbard model. 64

5.1 Demonstration of the swap network for the 1D model using a fragment size of six. 80
5.2 Demonstration of the swap network for the 2D model using a 1D fragment of size four. 81
5.3 Sections of the swap network for a 2D fragment. 83
5.4 Demonstration of the measurement pattern for the 1D model using a fragment of size six. 85
5.5 Measurement of all the hopping terms between two sets of qubits. 86
5.6 Comparison of the HV-min and -max ansätze for the 2D model. 90
5.7 Energy per site with site occupancy for the 1D model. 92
5.8 Double occupancy per site with U for the 1D model. 92
5.9 Energy and double occupancy per site for the 2D Hubbard model. 93
5.10 Energy per site with site occupancy for the 1D model using VQE with measurement. 94
5.11 Sweeping through the chemical potential. 96

6.1 Timing of the X-rotation gate in QSL and QuEST. 105
6.2 Timing of the CNOT gate in QSL and QuEST. 106
6.3 Comparison of the binary search and sorting methods of measurement sampling. 109
6.4 Timing of the number-preserving simulator using the phase gate. 112

A.1 Comparison of computational basis states and the ground state as the initial state for the NPr ansatz. 120
A.2 Comparison of the pre-initialised NPr ansatz and the standard NPr ansatz. 121

List of Tables

3.1 Circuit depths per layer of the EHV and NPr ansätze for various architectures. . . . 38
3.2 Number of circuit preparations needed to measure all of the Hubbard model terms for different JW orderings. . . . 42
3.3 Number of occupied orbitals corresponding to the lowest energy of the $t = 1, U = 2$ Hubbard Hamiltonian. . . . 48
3.4 Infidelity reached using CD and SPSA optimisers with realistic measurements. . . . 56
3.5 Infidelities reached for simulations with depolarising noise, with and without error detection. . . . 57

5.1 Number of layers of two-qubit gates required to implement one layer of the ansatz, and circuit preparations needed to measure all of the terms in the embedded Hamiltonian. . . . 78
5.2 Parameter counts for one layer of the HV-min and -max ansätze. . . . 88
5.3 Depth of the ansatz required to achieve 1% relative error against the ground energy per site for the 1D model. . . . 89
5.4 Depth of the ansatz required to achieve 1% relative error against the ground energy per site for the 2D model. . . . 89

6.1 List of quantum computing simulators partially or fully coded in C/C++. 100
6.2 Amount of memory required to store the state vector. . . . 102
6.3 List of lookup tables required for all one- and two-qubit number preserved gates. . . . 113

List of Abbreviations

1-RDM one-particle reduced density matrix.

CD coordinate descent.

DMET density matrix embedding theory.

EHV efficient Hamiltonian variational.

HV Hamiltonian variational.

JW Jordan-Wigner.

L-BFGS limited-memory Broyden-Fletcher-Goldfarb-Shanno.

NISQ noisy intermediate-scale quantum.

NPr number-preserving.

SPSA simultaneous perturbation stochastic approximation.

UCC unitary coupled-cluster.

VQA variational quantum algorithm.

VQE variational quantum eigensolver.

Chapter 1

Introduction

The field of quantum computing is currently undergoing a period of intense research. The construction of devices with more than 50 qubits [1, 2] has brought closer the promise of being able to solve classically intractable problems. Quantum computers take advantage of the properties of quantum systems such as superposition and entanglement to speed up solutions to certain types of problems. Famous examples are Shor's algorithm for factorising large numbers [3], Grover's algorithm for unstructured search [4] and, naturally, the simulation of quantum systems [5].

Current quantum devices are "noisy"; controlling a quantum system is a difficult task and current technology is not capable of maintaining the states of the qubits for very long. Nevertheless, the race is on to reach "quantum advantage", the point at which a useful task (beyond the ability of classical computers) can be carried out on a quantum computer. To reach a quantum advantage we not only need research on the hardware for quantum computing, but also on appropriate algorithms and applications that make best use of the limited resources of near-term quantum computers.

Applying variational quantum algorithms (VQAs) to quantum chemistry problems is expected to be one of the ways quantum advantage will be reached [6, 7]. Analogously to machine learning, VQAs prepare a parametrised quantum circuit and use a classical optimiser to find the parameters that best solve the problem at hand; they are also known as hybrid quantum-classical algorithms. The hybrid nature of these algorithms places a looser requirement on qubit coherence times and they have also been shown to have some resilience to noise, making them suitable for use on current and near-term quantum computers [8, 9].

Quantum chemistry problems are a good target application due to their exponential nature. Solving an electronic structure problem on N spin orbitals requires the storage of 2^N complex amplitudes on a classical computer, but only N qubits on a quantum device. We cannot simulate 50 qubits using a powerful supercomputer, but it seems reasonable that it may be possible to do a useful 50-qubit computation on a quantum device in the next couple of years.

There are a lot of aspects of VQAs that can be tailored to the specific problem being solved. For example, the mapping of the fermionic quantum chemistry problem onto qubits, the precise parametrised circuit run on the quantum computer and the classical optimisation routine. Systematic and thorough studies involving high-performance classical simulations are required to assess the performance of these algorithms. The aim of this thesis is to perform this investigation for a VQA called the variational quantum eigensolver (VQE) applied to the Hubbard model. The Hubbard model [10] is one of the simplest models of interacting electrons in a grid, making it a good test case for trying out techniques before extending to more complicated systems. The 2D model has remained unsolved despite decades of research and is thought to be relevant to applications such as high-temperature superconductivity [11].

In the remainder of this introductory chapter, we will define this problem in more detail. We will begin by giving a brief overview of the key aspects of quantum computing in Section 1.1. We will then discuss noisy quantum devices, the restrictions they place on the types of computations that can be run, and VQAs in Section 1.2. Following this, in Section 1.3 we formally define the Hubbard model and explain what makes it an interesting model to solve on a quantum computer. We conclude by giving an outline of the rest of the thesis in Section 1.4.

1.1 Quantum computing

The quantum analogue of the classical bit is the quantum bit, or qubit. Whereas bits can only be 0 or 1 at any time, a qubit can be in a superposition (i.e. linear combination) of the two. As such, a qubit $|\psi\rangle$ exists in the two-dimensional complex Hilbert space $\mathbb{C}^2$ and can be written as

$$|\psi\rangle = a|0\rangle + b|1\rangle \equiv \begin{pmatrix} a \\ b \end{pmatrix}, \tag{1.1}$$

where $a, b \in \mathbb{C}$. When the qubit is measured, it collapses to one of the states $|0\rangle$ or $|1\rangle$. The outcome ‘0’ is observed with probability $|a|^2$ and the outcome ‘1’ with probability $|b|^2$. The condition $|a|^2 + |b|^2 = 1$ is imposed so that the probabilities sum to one. The dual space associated to the Hilbert space consists of dual vectors of $|\psi\rangle$, denoted

$$\langle\psi| = a^*\langle 0| + b^*\langle 1| \equiv \begin{pmatrix} a^* & b^* \end{pmatrix}, \tag{1.2}$$

where a^* is the complex conjugate of a.

To represent multi-qubit systems, we take the tensor product of their Hilbert spaces. For example, $|\psi_1\rangle \otimes |\psi_2\rangle$ exists in the four-dimensional Hilbert space $\mathbb{C}^2 \otimes \mathbb{C}^2$. In general, a quantum state of n qubits exists in a 2^n-dimensional Hilbert space. This is partly where the power of quantum computing comes from, an n-qubit quantum computer can be in a superposition of up to 2^n states at once, whereas a classical computer can be in only one of those states.

An important aspect of quantum mechanics is entanglement. Two or more qubits are entangled if they cannot be factored into a tensor product of states of its component systems. For instance, if a two-qubit state $|\psi\rangle$ can be written as $|\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle$ where $|\psi_1\rangle, |\psi_2\rangle \in \mathbb{C}^2$, then it is a product state; otherwise it is an entangled state.

To preserve the norm of the quantum state vector in a Hilbert space, the linear operators that act on qubits must be unitary. A unitary operator U is defined by $UU^\dagger = U^\dagger U = I$ where I is the identity operator and $U^\dagger$ is the adjoint (conjugate transpose) of U. Due to this property of unitary operators, all quantum computations are reversible, as the adjoint of the operation done can be applied to reverse it. For an n-qubit system, the Hilbert space is 2^n-dimensional, so all unitary operators can be expressed as $2^n \times 2^n$ complex matrices.

In the language of quantum computation, we call these unitary operations gates. The most common types of gates are one- or two-qubit gates. An important group of one-qubit gates are the Pauli matrices

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{1.3}$$

The gate X corresponds to a bit-flip, swapping between $|0\rangle$ and $|1\rangle$. The gate Z corresponds to a phase-flip, adding a phase of -1 to $|1\rangle$ and leaving $|0\rangle$ unchanged. Furthermore, the exponentials of these matrices, for example $R_X(\theta) = e^{-i\theta X/2}$, are rotation matrices, which are another commonly used group of gates. An arbitrary one-qubit gate can be constructed using these rotation matrices as $U = R_X(\alpha)R_Z(\beta)R_X(\gamma)$ [12].

Two-qubit gates are typically used to create entanglement. An example of a two-qubit gate is the controlled-NOT (CNOT) operation. This gate applies an X

gate to the second qubit if the first qubit is in the state $|1\rangle$:

$$\text{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}. \tag{1.4}$$

It can be shown that any n-qubit unitary can be constructed using combinations of arbitrary one-qubit and CNOT gates, making this set of gates universal for quantum computation [12].

A quantum computation will typically start by preparing all the qubits in the $|0\rangle$ state and then applying gates to construct a quantum circuit. The final step is measuring the qubits in the computational basis to learn something about the quantum state that has been produced at the end of the computation. This can require multiple preparations of the quantum state, as each time we measure, the state collapse to one of the 2^n basis states of the Hilbert space.

1.2 Noisy intermediate-scale quantum algorithms

The term "noisy intermediate-scale quantum" (NISQ) was coined by Preskill in 2018 to describe the regime where quantum computers may be able to surpass classical computers, but are still too small for fault-tolerance [13]. "Noisy" refers to the fact that qubits in this regime will likely be low quality and subject to many different types of error, some of which will be discussed below. "Intermediate-scale" refers to the size of such devices, ranging from around 50 to a couple of hundred qubits. Fifty qubits is well beyond what can be classically simulated on the world's largest supercomputers, when storing the entire state vector and doing an arbitrary computation[1].

We will now discuss some of the characteristics that NISQ devices are expected to have, and the consequences for the types of computations that can be performed.

- **Qubit decoherence** – Controlling a quantum system is a difficult task. Current technology is not capable of maintaining very long coherence times [16,

---

[1] An arbitrary 50-qubit simulation would require 8 Petabytes of RAM using single-precision floating-point arithmetic. However, simulations of more qubits using less memory have been demonstrated using tensor networks for specific types of circuits [14, 15].

17]; this restricts the depth of the circuit that can be run on the device before the qubits decohere into the maximally mixed state.

- **Gate errors** – On a NISQ device it will likely not be possible to perform a quantum gate without introducing some error into the computation; this limits the number of gates that can be carried out on the qubits. Two-qubit gate fidelities of over 99% for superconducting qubits [16] and as high as 99.9% for trapped-ions [17] have been reported, but this still restricts circuits to having around 1000 gates [13].

- **Limited connectivity** – Ideally, we would like to be able to apply two-qubit gates between arbitrary pairs of qubits. A device with this property is said to be fully-connected. In reality, it is not always possible to construct a fully-connected device. For example, due to the way superconducting qubits work, qubits have to be physically close on the device to be able to perform two-qubit gates between them. This limited connectivity means that qubits may need to be swapped around in the quantum circuit, increasing the circuit depth (see Section 3.1.3). In addition, the variety of different connectivities such as Google's offset-rectangular lattices [1], Rigetti's square-octagon lattice [18] and IBM's connected rectangles [19], mean that quantum circuits have to be re-designed for different quantum devices.

- **Limited gate sets** – Different quantum devices have different types of gates that can be done "natively" on the hardware. Before being run, an arbitrary quantum gate will need to be decomposed in terms of these one- and two-qubit native gates to run on the hardware; this affects the gate depth of the circuit. NISQ devices will likely have a small native gate set, thereby increasing the gate depth required to run arbitrary circuits. For example, the iSWAP and controlled-phase gates are some of the two-qubit gates that are native to superconducting hardware [1, 16]. To do a CNOT gate – a basic building block of quantum algorithms – two iSWAP gates and several one-qubit gates need to be strung together.

- **No error correction** – One of the key aspects of NISQ devices is that there will be no in-built error detection or correction. Errors caused by decoherence, applying gates, or measurement readout will need to be taken into consideration manually when running a quantum circuit. On NISQ devices it will be advantageous to run algorithms which have a resilience to noise, whilst also making use of error mitigation techniques [20].

To make the most out of NISQ devices, we must carefully design the algorithms and circuits that will be run on them. A class of algorithms which have shown significant promise are VQAs, also known as hybrid quantum-classical algorithms. The field of VQAs is already too large to summarise here; see [8, 9, 20] for reviews. A wide variety of algorithms and applications have been researched, such as algorithms to variationally solve systems of linear equations, factorise numbers and simulate Hamiltonians.

One popular VQA is the quantum approximate optimisation algorithm (QAOA) which can be used to solve combinatorial problems such as MaxCut [21]. In this thesis, we focus on the algorithm which kick-started the field, the VQE. It was developed by Peruzzo et al. in 2013 to calculate the ground state energies for molecules [22]. Since then, adaptations of the VQE algorithm have been introduced to also find excited states [23–25], which could be useful for investigating chemical reaction dynamics. VQE can also be extended to the study of open systems which are governed by Lindblad operators. Mapping a density matrix onto the quantum computer doubles the number of qubits, but being able to simulate quantum systems that interact with the environment could, for example, facilitate the investigation of non-equilibrium states [26]. In this thesis, however, we restrict the work to the standard VQE algorithm for calculating ground states. Chapter 2 contains an in-depth overview of the standard VQE algorithm, covering many of the details of VQAs that are omitted in this brief introduction.

VQAs are suitable NISQ algorithms for a number of reasons. First, the hybrid quantum-classical nature of these algorithms keeps the circuit depth low, which is necessary due to the short coherence times of the qubits. For instance, the circuit requirements for implementing the more traditional quantum phase estimation algorithm are much higher than using the VQE to find the ground state of a Hamiltonian [27]. Second, VQAs are naturally able to suppress some forms of noise. For example, systematic coherent errors such as over-rotations can be corrected by the classical optimiser as the specific values of the parameters are irrelevant [8, 28]. Third, there is flexibility in the parametrised quantum circuits that are run as part of the algorithms. It is possible to tailor these circuits to the device, making use of its specific strengths (see Section 2.3).

On the other hand, VQAs can suffer from the problem of "barren plateaus" where the gradient of the objective function decays exponentially to zero as the number of qubits increases. These barren plateaus appear when running randomly initialised random circuits and prevent the classical optimisation component of the VQA from working [29]. In addition, hardware noise of the types described above have been shown to cause barren plateaus, even when running more structured circuits and when using gradient-free optimisation routines. These noise-induced barren plateaus negatively impact the scalability of VQAs on NISQ devices [30].

Certain problems which VQAs attempt to solve can be shown to be computationally hard. For example, finding the ground state energy of an arbitrary k-local Hamiltonian is QMA-complete [31], even for instances of simple models such as the Hubbard model [32]. This means, in general, that a quantum algorithm such as the VQE cannot be guaranteed to find the ground state of a Hamiltonian in polynomial time. In addition, even if we are trying to find the ground state of a Hamiltonian that is not computationally hard, the VQA optimisation problem may still be NP-hard [33]. However, simple problems in machine learning have similarly been shown to be NP-hard [34], but this does not mean that no machine learning algorithm is viable. Furthermore, a large amount of numerical studies and experiments that have been carried out on quantum hardware have shown that VQAs can work in practice for small system sizes [35–37].

Investigating VQAs theoretically is difficult and it is often not possible to prove rigorous results about them. For certain toy models, the landscape of the optimisation function has been investigated, but in general is non-convex and very complicated, making it difficult to prove that the algorithm will converge [38]. As a result, numerical work is the most popular approach to investigating the performance of VQAs. Although there may be some limits to the applicability of numerical simulations outside the region of validity (e.g. > 30 qubits), it is a viable approach to show what might be suitable on larger NISQ devices.

1.3 The Hubbard model

The Hubbard model was independently formulated in the 1960s by Gutzwiller [39], Hubbard [10] and Kanamori [40] to describe the behaviour of strongly correlated electrons in transition metals. Since then, the Hubbard model has been found to describe a wide variety of physical phenomena such as metal-insulator transitions, superconductivity, magnetism and topologically ordered phases [11, 41, 42].

In the Hubbard model, the atoms in a solid are simplified to a lattice of sites where each site is a single spatial orbital; i.e. the site may be empty, occupied by a spin-up electron, occupied by a spin-down electron, or doubly occupied by both spin-up and -down electrons. Electrons can move between these sites and double-occupied sites are subject to an energy penalty. The Hubbard Hamiltonian is defined as

$$H_{\text{hub}} = -\sum_{ij,\sigma} t_{ij}(a^{\dagger}_{i\sigma}a_{j\sigma} + a^{\dagger}_{j\sigma}a_{i\sigma}) + U\sum_{i} n_{i\uparrow}n_{i\downarrow}, \tag{1.5}$$

where $a^{\dagger}_{i\sigma}$, $a_{i\sigma}$ are fermionic creation and annihilation operators for a spin-$\sigma \in \{\uparrow,\downarrow\}$ electron in site i; $n_{i\sigma} = a^{\dagger}_{i\sigma}a_{i\sigma}$ is the number operator. The first term in equation (1.5) is called the "hopping term", as it represents spin-σ fermions hopping between sites i and j. Typically (and in this thesis) the nearest-neighbour Hubbard model is considered, meaning that we take $t_{ij} = t$ for sites that are adjacent in the lattice and $t_{ij} = 0$ elsewhere. The parameter t is then called the "tunnelling amplitude". The second term is called the interaction or "onsite term", where U is the "Coulomb potential".

More specifically, in this thesis we will focus on solving the 1D and 2D Hubbard models occupied by N_{occ} electrons. For the 1D model, we consider a line of sites of length N. The 2D model uses a rectangular lattice ("grid") of $N = N_x \times N_y$ sites. An example of this is demonstrated in Figure 1.1 for a 3×3 grid with nearest-neighbour hopping interactions. The nearest-neighbour terms correspond to hopping terms between horizontally and vertically adjacent sites[2]. When periodic boundary conditions are considered, there are additional hopping terms between sites in the first and last rows, and the first and last columns, of the grid. The observables that we are interested in calculating in this thesis are the ground state energy and the double occupancy. The double occupancy per site is calculated as $\frac{1}{N}\sum_i \langle n_{i\uparrow}n_{i\downarrow}\rangle$ and measures the probability of finding two electrons on one site.

While the 1D Hubbard model is exactly solvable using the Bethe ansatz [43, 44], a full description of the 2D model and its phase diagram is still an open problem. In the $U \gg t$ limit at half-filling (one electron per site), the Hubbard model reduces to the antiferromagnetic Heisenberg model; away from half-filling, it reduces to the $t - J$ model [45]. In this thesis we will focus on the case of $U/t = 1$ to 8, known as the weak coupling and intermediate-to-strong coupling regimes, where the Hubbard model is thought to exhibit insulating, magnetic, superconducting and striped behaviours. A wide variety of numerical methods have been used to probe the Hubbard model in these parameter regimes [46–48].

---

[2] A common extension is to consider next-nearest-neighbour hopping terms. These would correspond to hopping interactions between diagonally adjacent sites with $t_{ij} = t'$.

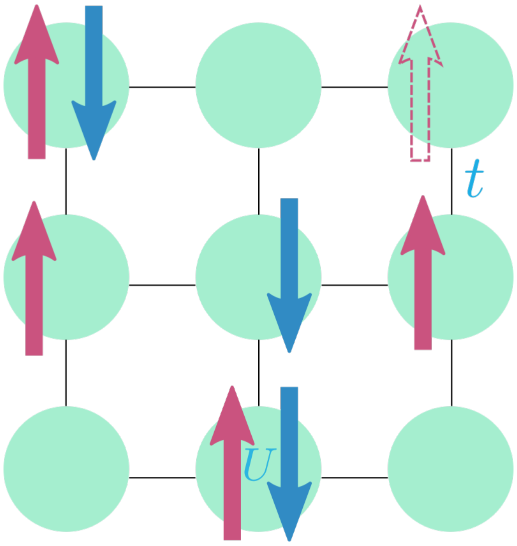


Figure 1.1: The Hubbard model on a 3×3 grid occupied by four spin-up and three spin-down electrons. This is a demonstration of nearest-neighbour hopping and onsite terms.

Despite it's apparent simplicity, the properties of the Hubbard model are far from fully understood [11, 46, 47]. Its regular structure and relatively simple form suggests it may be easier to implement on a NISQ device than, for example, model systems occurring in quantum chemistry. This makes the Hubbard model a good test bed for trying out different techniques, before extending to more complex models. At the same time, the use of near-term quantum computers may be able to reveal properties of the Hubbard model of relevance to problems of practical importance, such as high-temperature superconductivity [49].

On a grid with N sites, the Hubbard Hamiltonian can be represented as a sparse square matrix with 2^{2N} rows and columns. Although the size of this matrix can be reduced by restricting to a subspace corresponding to a given occupation number, and taking advantage of translation- and spin-invariance, the worst-case growth of the size of these subspaces is still exponential in N. This exponential growth severely limits the capability of classical exact solvers to address this model. For example, Yamada et al. [50] report an exact solution of the Hubbard model with 17 fermions on 22 sites requiring over 7TB of memory and 13 TFlops on a 512-node supercomputer. By contrast, a Hubbard model instance with N sites can be represented using a quantum computer with $2N$ qubits (each site can contain at most one spin-up and at most one spin-down fermion, so two qubits are required per site). This suggests that a quantum computer with around 50 qubits could already simulate instances of the Hubbard model beyond classical capabilities.

Approximate classical techniques such as the quantum Monte Carlo (QMC) and density matrix renormalisation group (DMRG) methods can address larger grids than near-term quantum computers (up to thousands of sites), but experience difficulties in stronger coupling regimes and away from half-filling, leading to substantial uncertainties in physical quantities [46]. Another approach to understanding the Hubbard model via a quantum device is analogue quantum simulation [51, 52]: engineering a special-purpose quantum system that implements the Hubbard Hamiltonian directly [53–55]. While these analogue simulators are easier to implement experimentally than universal quantum computers, and enable access to much larger systems than will be possible using near-term quantum computers, they are inherently less flexible than digital quantum simulation. The hope is that quantum computing could evade the difficulties experienced by these various methods and enable a more direct solution to the Hubbard model.

1.4 Thesis outline

The focus of this thesis is on assessing the performance of the VQE algorithm when solving the Hubbard model. We begin in Chapter 2 by giving a detailed overview of the VQE algorithm. We discuss fermion-to-qubit encodings, ansatz circuits, measurement of expectation values and the classical optimisation routine. In each case, we give a brief summary of the field and then focus on techniques that we use Chapters 3 and 5. We discuss the application of VQE to a general quadratic Hamiltonian with Hubbard-like onsite interactions, which covers the types of Hamiltonians solved in this thesis.

In Chapter 3 we present a systematic study into finding the ground state of the Hubbard model using VQE. We first introduce a generalisation of the Hamiltonian variational ansatz called the number-preserving ansatz, which gives more freedom in the choice of initial state. We determine the precise complexity of implementing this ansatz circuit for fully-connected, nearest-neighbour and Google Sycamore device architectures. We then carry out extensive numerical simulations for grids with up to 12 sites (24 qubits) at three levels of realism: assuming that we know the exact expectation value to test which ansätze are effective; simulating realistic measurements; and simulating depolarising noise to test how well the classical optimisers cope with varying types and levels of noise. Based on the results, it seems plausible that an instance of the Hubbard model larger than the capacity of classical exact diagonalisation methods could be solved by optimising over quantum circuits with depth 300–500 (on a fully-connected architecture). This is substantially smaller than previous estimates for other proposed applications of near-term quantum computers, albeit beyond the capacity of leading hardware available today.

Calculating the ground state properties of a Hamiltonian can be mapped to the problem of finding the ground state of a smaller Hamiltonian through the use of embedding methods. These embedding techniques have the ability to drastically reduce the problem size, and hence the number of qubits required when running on a quantum computer. However, the embedding process can produce a relatively complicated Hamiltonian, leading to a more complex quantum algorithm. In Chapters 4 and 5, we investigate how one of these embedding techniques, density matrix embedding theory (DMET) could be implemented on a quantum computer to solve the Hubbard model. In particular, we consider the VQE algorithm as the solver for the embedded Hamiltonian within the DMET algorithm.

In Chapter 4 we focus on the "classical" aspects of the combined DMET and VQE algorithm. We begin by introducing the DMET algorithm. This is followed by an in-depth description of how the single-shot embedding algorithm (a variant of DMET) can be applied to the Hubbard model; we have found this to be lacking in other literature. We conclude the chapter with a derivation of the structure of the embedded Hamiltonian produced by DMET, which is key to determining the complexity of the VQE algorithm.

Chapter 5 covers the "quantum" aspects of the combined DMET and VQE algorithm. We carry out a similar analysis to the one in Chapter 3 by determining detailed resource and complexity estimates for running the VQE algorithm on a fully-connected quantum device. The theoretical work is followed by numerical simulations up to a fragment size of four (16 qubits) for a variety of Hubbard model parameters. We find that DMET with VQE is an efficient and accurate method for finding ground state properties of the Hubbard model. In our experiments, we were able to reproduce previous results based on exact diagonalisation.

The results presented in this thesis, in particular in Chapters 3 and 5, rely greatly on being able to run many/large VQE simulations in a reasonable time frame. To do this we wrote code in C++ using the Quantum Exact Simulation Toolkit (QuEST) [56] to simulate the quantum circuits, which we ran on GPUs. In Chapter 6, we present a new quantum simulator coded in C++ called the Quantum Simulation Library (QSL), intended to speed up the type of simulations performed in this thesis. We discuss the techniques and methods used in the development of

QSL, for example the best way to lay out the state vector, and how to simulate gates and measurements as efficiently as possible. Timings are compared against QuEST and we find that our simulator can implement gates up to 8-10× faster. We conclude by showing how known properties of a circuit, such as number or spin preservation, could be used to speed up simulations. This could be useful when simulating the VQE algorithm for quantum chemistry problems.

1.4.1 List of publications

Chapter 3, Appendix A and parts of Chapter 2 and Section 1.3 are based on the following paper:

- "Strategies for solving the Fermi-Hubbard model on near-term quantum computers" by Chris Cade, Lana Mineh, Ashley Montanaro and Stasja Stanisic. *Physical Review B* 102, 235122 (2020) [57].
  Data are available at the University of Bristol data repository, data.bris [58].

Chapters 4, 5, Appendix B and parts of Chapter 2 are based on:

- "Solving the Hubbard model using density matrix embedding theory and the variational quantum eigensolver" by Lana Mineh and Ashley Montanaro. *Physical Review B* 105, 125117 (2022) [59].
  Data are available at the University of Bristol data repository, data.bris [60].

Chapter 6 is based on unpublished work done whilst coding the following quantum simulator:

- C++ Quantum Simulation Library (QSL). Lana Mineh and John R. Scott `https://github.com/lanamineh/qsl` [61].

The following work was published during the course of the PhD, but is outside of the scope of this thesis:

- "Designing quantum experiments with a genetic algorithm" by Rosanna Nichols, Lana Mineh, Jesús Rubio, Jonathan C. F. Matthews and Paul A. Knott. *Quantum Science and Technology* 4(4), 2019 [62].

Chapter 2

The variational quantum eigensolver

The VQE is a hybrid quantum-classical algorithm used to produce the ground state of Hamiltonians. As mentioned in Section 1.2, it is suitable as a NISQ algorithm because of lower circuit depth requirements (as opposed to an algorithm such as phase estimation) and intrinsic robustness to certain types of noise [28]. The VQE algorithm was developed by Peruzzo et al. in 2013 as a way of using a quantum co-processor to calculate ground state energies for molecules [22]. The initial experiment was done on a photonics chip, but since then the VQE has been implemented using ion traps and superconducting qubits [27, 35, 63, 64]. The VQE algorithm can potentially be tailored to run on any quantum hardware.

This chapter is intended as an overview of the VQE algorithm, with a particular focus on techniques that will form a basis for Chapters 3 and 5. Parts of this chapter have appeared in "Strategies for solving the Fermi-Hubbard model on near-term quantum computers" [57] and "Solving the Hubbard model using density matrix embedding theory and the variational quantum eigensolver" [59]. Sections 2.4 and 2.5.2 contain original work that was done during the course of these projects.

2.1 Outline of the VQE algorithm

The VQE algorithm used to produce the ground state of a Hamiltonian H. It relies on the variational principle, which states that

$$\langle\psi|H|\psi\rangle \geq E_g, \tag{2.1}$$

where $|\psi\rangle$ is an arbitrary normalised state and E_g is the ground energy (lowest eigenvalue) of H.

In brief, the steps of the algorithm are [8, 28]:

1. Prepare a parametrised state $|\psi(\boldsymbol{\theta})\rangle = U(\boldsymbol{\theta})|\psi_0\rangle$ on the quantum computer. $U(\boldsymbol{\theta})$ is an ansatz circuit intended to reproduce the ground state and $|\psi_0\rangle$ is an initial state.

2. Measure the expectation value $\langle\psi(\boldsymbol{\theta})|H|\psi(\boldsymbol{\theta})\rangle$. Multiple preparations of $|\psi(\boldsymbol{\theta})\rangle$ on the quantum computer are required to accurately estimate this expectation value.

3. Use a classical optimisation method to determine a new value for $\boldsymbol{\theta}$ that will minimise the expectation value.

4. Repeat steps 1-3 until the optimiser converges. The final value of $\boldsymbol{\theta}$ will parameterise the ground state and give an expectation value equal to the ground energy.

We will spend the rest of this chapter explaining these steps in detail. At each step of the algorithm there are many choices that can be made. For example, it is desirable to choose an ansatz circuit that can parameterise the ground state in as few parameters and as low a circuit depth as possible. In addition, we would like to measure the expectation value using the minimum number of circuit preparations. We must also carefully consider the classical part of the algorithm; we need to choose a suitable optimisation algorithm that typically converges in a low number of iterations and can cope with noise. The VQE algorithm performs best if all of these are tailored to the Hamiltonian H that is being solved.

In this thesis we are interested in the application of the VQE algorithm to fermionic Hamiltonians of the form

$$H_F = \sum_{i \neq j, \sigma} t_{ij}\,(a^\dagger_{i\sigma} a_{j\sigma} + a^\dagger_{j\sigma} a_{i\sigma}) + \sum_i U_i\, n_{i\uparrow} n_{i\downarrow} + \sum_{i,\sigma} \mu_i\, n_{i\sigma}, \tag{2.2}$$

where t_{ij}, U_i, $\mu_i \in \mathbb{R}$. As for the Hubbard model in Section 1.3, we will refer to terms in the first sum as hopping terms, as they represent fermions of spin-σ hopping between spatial orbitals (sites) i and j. Terms in the second sum will be called onsite terms as they represent the interaction between a spin-up and -down fermion occupying the same site. The terms in the final sum will be referred to as number terms as $n_{i\sigma}$ is the fermionic number operator.

H_F is a general quadratic Hamiltonian with Hubbard-like onsite interactions, it preserves both electron number and spin. This covers both the Hubbard model (which will be solved directly in Chapter 3) and the embedded Hamiltonian obtained from DMET (see Chapters 4 and 5). Some of the discussion in this chapter will be restricted to Hamiltonians of this form; for example in Section 2.2 we will focus on how to encode H_F on a quantum computer, in Section 2.3.1 we show how to apply the Hamiltonian variational ansatz to H_F and in Section 2.4 we discuss how to measure the expectation value $\langle H_F \rangle$.

2.2 Fermion-to-qubit encodings

When using the VQE to find the ground state of a fermionic Hamiltonian, we must first transform it to a qubit (spin-$\frac{1}{2}$) Hamiltonian using a fermion-to-qubit encoding. This transformation is necessary as we need a way of mapping operators that act on indistinguishable fermions to distinguishable qubits. Fermion-to-qubit encodings preserve the canonical anti-commutation relations

$$\{a_i^\dagger, a_j\} = \delta_{ij}, \quad \{a_i^\dagger, a_j^\dagger\} = \{a_i, a_j\} = 0. \tag{2.3}$$

In this thesis we use the Jordan-Wigner (JW) encoding [65], which is one of the simplest encodings. It introduces no overhead in qubit count as each spin orbital[1] maps to one qubit. The state $|0\rangle$ represents an unoccupied orbital and $|1\rangle$ occupied. The JW encoding is given by

$$a_j^\dagger \mapsto \frac{1}{2}(X_j - iY_j)Z_0 \cdots Z_{j-1} \tag{2.4}$$

$$a_j \mapsto \frac{1}{2}(X_j + iY_j)Z_0 \cdots Z_{j-1}, \tag{2.5}$$

where X_j, Y_j and Z_j are the Pauli operators acting on qubit j. The JW encoding requires us to choose an ordering for the spin orbitals as it maps the fermionic modes to a line of qubits. We will often have an ordering where all of the spin-up orbitals are followed by the spin-down.

Turning to the fermionic Hamiltonian from equation (2.2), the hopping terms between qubits are transformed as

$$a_j^\dagger a_k + a_k^\dagger a_j \mapsto \frac{1}{2}(X_j X_k + Y_j Y_k) Z_{j+1} \cdots Z_{k-1}, \tag{2.6}$$

---

[1] A spin orbital combines the site i and spin σ indexing into one. Spin orbitals can be either occupied by an electron or unoccupied, unlike spatial orbitals which can have four different configurations.

where $j < k$ without loss of generality. The number operator terms become

$$n_j = a_j^\dagger a_j \mapsto \frac{1}{2}(I - Z_j) = |1\rangle\langle 1|_j, \tag{2.7}$$

where I is the identity matrix, and the onsite terms become

$$n_j n_k \mapsto \frac{1}{4}(I - Z_j)(I - Z_k) = |11\rangle\langle 11|_{jk}. \tag{2.8}$$

The hopping terms contain long Z-strings, leading to encoded operators with a large Pauli weight (the number of qubits the Pauli string acts on). If we wanted to apply an evolution according to this term on a quantum computer i.e. $e^{i(a_k^\dagger a_j + a_j^\dagger a_k)}$, this would be a gate on $|j-k|+1$ qubits which can be costly to implement (see Section 2.3.1). This makes the ordering of the orbitals important; certain orderings can either reduce or increase the length of the Z-strings. Fermionic swap networks [66] for implementing hopping term evolutions can also help mitigate issues caused by these long Z-strings (see Section 2.3.2).

There are a variety of other fermion-to-qubit encodings which produce operators that have a lower Pauli weight, but at a cost of additional qubits [67–69]. There are also encodings which prioritise decreasing the number of qubits at the expense of more complicated or additional quantum gates [70]. Examples of the former are the Ball-Verstraete-Cirac (BVC) and Bravyi-Kitaev superfast (BKSF) encodings. The BVC encoding [71, 72] adds an extra auxiliary qubit for every qubit (doubling the qubit count). These auxiliary qubits are then used to carry out the hopping interactions, eliminating the need for Z-strings. The BKSF encoding [73, 74] introduces a qubit for every interaction term in the Hamiltonian and produces Pauli weights of $O(d)$, where d is degree of the Hamiltonian's interaction graph.

The compromise between having a low qubit count versus low Pauli weights is one that should be carefully considered based on the Hamiltonian to be solved and the targeted quantum device. In the regime of the NISQ era where there are a lower number of qubits, it may be preferable to use the JW encoding to maximise the size of the Hamiltonians solved. It may also be more efficient to use a simpler encoding. An example of this is the $N_x \times N_y$ Hubbard model; the JW encoding is more efficient in terms of both qubit count and circuit depth (compared to the BVC and BKSF encodings) when $\min(N_x, N_y) \leq 8$ [57]. On the other hand, it has been shown that the long Z-strings produced by the JW encoding can lead to barren plateaus in VQAs, potentially limiting the use of JW for larger systems [75, 76].

2.3 Ansatz circuits

The ansatz circuit is the cornerstone of the VQE algorithm. The circuit $U(\boldsymbol{\theta})$ is used to prepare the parametrised state $|\psi(\boldsymbol{\theta})\rangle = U(\boldsymbol{\theta})|\psi_0\rangle$ on the quantum computer. An ideal ansatz circuit would parameterise part of the Hilbert space which contains the ground state of H, while having a low circuit depth and as few variational parameters as possible. Having fewer parameters simplifies the classical optimisation problem and a low circuit depth makes it suitable for use on NISQ devices.

When choosing an ansatz circuit we must keep in mind the specific problem that is being solved. For example, if we know our problem has certain symmetries or properties, then picking an ansatz that respects these will likely perform better than one that does not. On the other hand, if the circuit required for this is too complex, then it may not be easily implementable on a NISQ device. It is important to keep in mind that the goal is to be able to run the circuit on a NISQ computer, and consider the challenges this poses (e.g. restricted gate set, limited qubit connectivity).

Developing ansatz circuits is an active area of research; see [8, 9] for reviews. Below we present a summary of some popular ansatz circuits. In Section 2.3.1 we provide details about the Hamiltonian variational ansatz, which we will make use of in Chapters 3 and 5.

Hardware efficient ansatz

The hardware efficient ansatz was designed to be used on NISQ devices [35], therefore it is very generic and does not take any properties of the problem into account. Gates can be chosen from the native gate set of the hardware, and two-qubit gates only performed between qubits that are connected on the device.

The initial state for this ansatz is usually the all-zero state $|0\rangle^{\otimes n}$. The ansatz circuit itself typically consists of a series of parametrised one-qubit gates, followed by fixed two-qubit entangling gates. This structure is repeated D times, where D is the "ansatz depth" (see Figure 2.1). This repeating structure is typical of many ansatz circuits. Using Euler angles, an arbitrary one-qubit gate can be decomposed in terms of three rotation gates, e.g. $R_Z(\alpha)R_X(\beta)R_Z(\gamma)$, leading to a parameter count of $3nD$. For n qubits, an example of the hardware efficient ansatz circuit is

$$|\psi(\boldsymbol{\theta})\rangle = \prod_{d=1}^{D} U_{\text{ent}} \prod_{i=1}^{n} R_Z(\theta_1^{i,d}) R_X(\theta_2^{i,d}) R_Z(\theta_3^{i,d}) \, |0\rangle^{\otimes n}, \tag{2.9}$$

where U_{ent} is a unitary representing the two-qubit entangling gates performed in one layer of the ansatz.

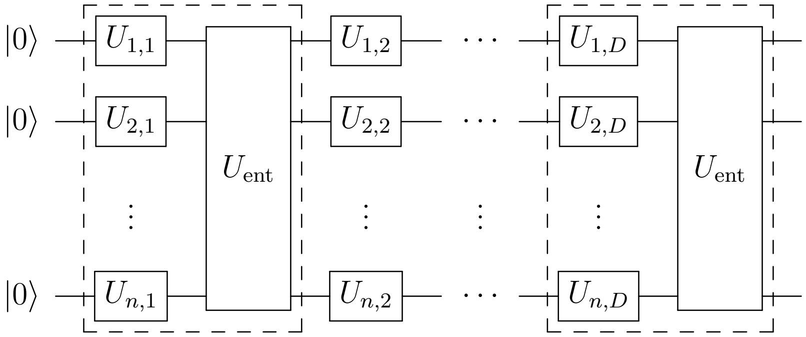


Figure 2.1: The hardware efficient ansatz circuit on n qubits. Arbitrary one-qubit gates $U_{i,d} = R_Z(\theta_1^{i,d})R_X(\theta_2^{i,d})R_Z(\theta_3^{i,d})$ act on the $|0\rangle^{\otimes n}$ state, followed by entangling gates that respect the connectivity of the device. This cycle repeats D times, where D is the ansatz depth.

The hardware efficient ansatz has been successfully used on quantum hardware for a low number of qubits (e.g. 2-6) [22, 35, 77, 78]. However, theoretical work has shown that these types of generic circuits suffer from the problem of barren plateaus, where the gradient of the objective function decays exponentially to zero as the number of qubits increases [29]. This could make the hardware efficient ansatz unsuitable for larger experiments.

Unitary coupled-cluster

Unitary coupled-cluster (UCC) is an example of a problem-inspired ansatz that is used for solving quantum chemistry problems. Coupled-cluster methods have been used for decades by the quantum chemistry community [79], making UCC a good candidate for an ansatz.

The UCC ansatz is defined by the following equation,

$$|\psi(\boldsymbol{\theta})\rangle = e^{T(\boldsymbol{\theta})-T^\dagger(\boldsymbol{\theta})}|\psi_0\rangle, \tag{2.10}$$

where $|\psi_0\rangle$ is often taken to be the Hartree-Fock ground state of H. The cluster operator $T(\boldsymbol{\theta}) = T_1(\boldsymbol{\theta}) + T_2(\boldsymbol{\theta}) + \dots$ is defined by

$$T_1(\boldsymbol{\theta}) = \sum_{i,j} \theta_{ij}\, a_i^\dagger a_j, \tag{2.11}$$

$$T_2(\boldsymbol{\theta}) = \sum_{i_1,i_2,j_1,j_2} \theta_{i_1 i_2 j_1 j_2}\, a_{i_1}^\dagger a_{j_1} a_{i_2}^\dagger a_{j_2}, \tag{2.12}$$

where the θ's are the variational parameters to optimise over, with the i's indexing the occupied orbitals and j's the unoccupied ones [28, 80].

The general idea is to explore states close to the initial state. For example, $T_1(\boldsymbol{\theta})$ is the single-excitation operator exploring the states where one of the electrons has moved from an occupied orbital to an unoccupied one. $T_2(\boldsymbol{\theta})$ is the double-excitation operator where two electrons have moved position. It is possible to include higher orders of $T_i(\boldsymbol{\theta})$, but going up to second order is usually sufficient for most quantum chemistry calculations. This version of UCC is often referred to as UCCSD (UCC with single- and double-excitations).

The circuit $e^{T(\boldsymbol{\theta})-T^\dagger(\boldsymbol{\theta})}$ cannot be implemented exactly on the quantum computer. To convert the UCC circuit to a sequence of one- and two-qubit gates, it is first Trotterised, leading to long circuits [80, 81]. For UCCSD, this leads to a gate scaling of $O(n^2m^3)$ using the JW encoding with $O(n^2m^2)$ variational parameters, where n is the number of spin-orbitals and m is the number of electrons in the system [80].

There have been many experiments implementing UCC on superconducting [27, 82, 83] and trapped-ion [63, 64, 84, 85] quantum computers. There have also been a number of variants of UCC developed which aim to make UCC more efficient to implement or more effective [86–88].

Other ansätze

It is possible to take an approach that is between the generic hardware efficient ansatz and the more specialised UCC. For example, Dallaire-Demers et al. introduce a low-depth ansatz which is inspired by quantum chemistry. Double-excitations are removed and replaced with nearest-neighbour phase coupling rotations, making it more viable for use on NISQ computers [87]. Gard et al. introduce an efficient ansatz which conserves particle number, total spin, spin projection, and time-reversal symmetry [89].

An entirely different approach is the one taken by ADAPT-VQE, where the form of the ansatz circuit itself is optimised over. The ansatz is grown gate by gate; these gates are kept or discarded depending on the performance of the ansatz circuit [90, 91].

2.3.1 The Hamiltonian variational ansatz

In this thesis we make use of the Hamiltonian variational (HV) ansatz [92]. This ansatz has been primarily studied in relation to solving the Hubbard model, where it has been shown to be effective [92–95]. It has also been used with small molecules [92], the transverse field Ising and XXZ models [96], and more recently the antiferromagnetic Heisenberg model on a Kagome lattice [97, 98]. While numerical evidence has shown that the HV ansatz may only have mild or non-existent barren plateaus [96], it may not be possible to avoid it for every problem [99].

The HV ansatz is inspired by adiabatic quantum state preparation and the QAOA algorithm [21]. Imagine we have a Hamiltonian $H = H_0 + H_1$, such that the ground state $|\psi_0\rangle$ of H_0 is easy to prepare. Using adiabatic state preparation, it is possible to reach the ground state of H from $|\psi_0\rangle$ by alternating many evolutions of the form e^{-itH_0}, e^{-itH_1} where the time step t is sufficiently small. The HV ansatz generalises this idea by allowing arbitrary angles in the evolution terms (as opposed to a fixed t) and splitting up the Hamiltonian into more than two terms,

$$H = \sum_j H_j, \tag{2.13}$$

where the terms inside each H_j are commuting.

The ansatz consists of applying evolutions of the form $e^{i\theta H_j}$ to an initial state, where θ is a parameter to be determined in the VQE optimisation loop. The initial state $|\psi_0\rangle$ is typically the ground state of the non-interacting part of H (for H_F this corresponds to $u_i = 0$). This is a quadratic Hamiltonian which means its ground state can be prepared efficiently on a quantum computer using Givens rotations [100]. The parametrised state is

$$|\psi(\boldsymbol{\theta})\rangle = \prod_d \prod_j e^{i\theta_{d,j} H_j} |\psi_0\rangle, \tag{2.14}$$

where applying all the H_j evolutions makes up one layer of the ansatz whose depth is indexed by d. The purpose of H_j is to group together terms which share the same variational parameter $\theta_{d,j}$. By only doing operations according to terms in the Hamiltonian, the HV ansatz can make use of properties of the Hamiltonian such as electron number and spin preservation.

It is possible to consider two extreme cases of splitting H into commuting groups H_j. In one case we can pick the groups so that there are as few as possible. In the other, each H_j can contain only one term from H which leads to the maximum number of parameters per layer. This has the effect of making the optimisation routine more difficult, but can reduce the ansatz depth required to solve the problem.

We will now discuss how the HV ansatz can be applied to H_F by laying out the specific implementation of $e^{i\theta H_j}$ in terms of quantum gates. There are three types of evolutions in equation (2.2): hopping terms, onsite terms and number operator

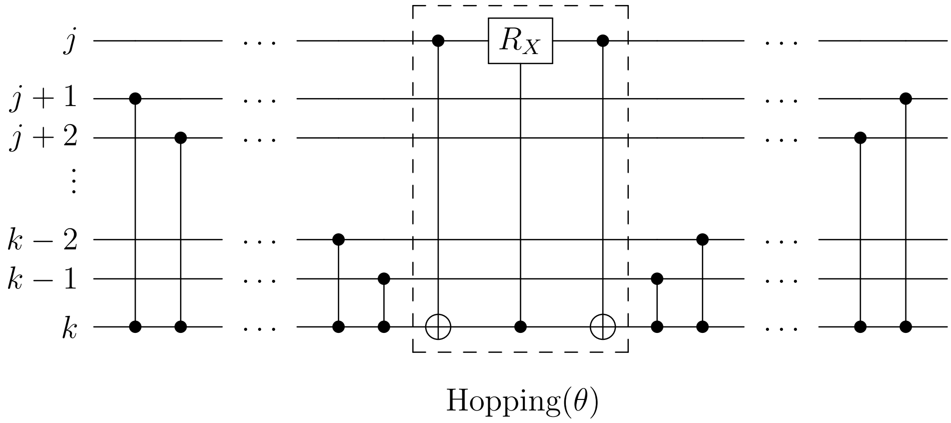


Figure 2.2: Implementation of the hopping gate between qubits j and k, decomposed as a series of two-qubit gates. The two-qubit hopping gate from equation (2.17) is sandwiched between a series of controlled-Z gates. Inside the box is an example decomposition of the hopping gate as three controlled gates – a controlled X rotation between two CNOTs.

terms. It is important that the terms in H_j commute so that the quantum circuit can be decomposed into these three types of operations.

The number terms can be implemented as

$$e^{i\theta n_j} = e^{i\theta|1\rangle\langle 1|_j} = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{pmatrix}, \tag{2.15}$$

which is a phase shift on qubit j. The onsite terms are implemented as

$$e^{i\theta n_j n_k} = e^{i\theta|11\rangle\langle 11|_{jk}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{i\theta} \end{pmatrix}, \tag{2.16}$$

which is a controlled phase shift between qubits j and k. The hopping terms are more complicated; starting with the case where j and k are neighbouring qubits in the JW ordering i.e. $k = j+1$,

$$e^{i\theta(a_j^\dagger a_k + a_k^\dagger a_j)} = e^{i\theta(X_j X_k + Y_j Y_k)/2} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & i\sin\theta & 0 \\ 0 & i\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{2.17}$$

We will refer to this gate as Hopping(θ); it performs an X rotation on the $\{|01\rangle, |10\rangle\}$ subspace. In general, the hopping gate $e^{i\theta(a_j^\dagger a_k + a_k^\dagger a_j)}$ is a $|j-k|+1$-qubit gate due to the Z-strings in the JW encoding. This can be decomposed as a series of $2|j-k|+1$ two-qubit gates [93] which is shown in Figure 2.2.

Since all the number terms commute with each other, we can, for example, group them altogether in one H_j and parameterise them all with a single θ. The same applies for the onsite terms. The hopping terms can then be split into a number of groups of commuting terms, depending on which t_{ij} are non-zero. In a typical HV ansatz circuit, at each layer of the ansatz we could implement all of the onsite gates, followed by the hopping gates and then the number gates.

2.3.2 Fermionic swap networks

The need for a relatively large numbers of gates to implement hopping gates motivates the use of fermionic swap networks [66]. We use fermionic swap (FSWAP) gates to move orbitals that were originally not adjacent in the JW ordering into JW-adjacent positions. The FSWAP gate acts as a swap gate for fermions (i.e. it preserves anti-symmetry), and corresponds to the unitary operator

$$\mathrm{FSWAP} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}. \tag{2.18}$$

Now, for example, if qubits j and k are swapped so that they are adjacent to each other in the JW ordering, the hopping gate between them can be implemented more simply as the two-qubit gate given by equation (2.17).

Using a swap network, only n layers of two-qubit gates are required to implement all possible hopping terms for n qubits. Figure 2.3 demonstrates how this can be done for four qubits. The swap network is formed of two alternating layers: in the first layer, odd-numbered qubits are swapped with the qubits to their right; and in the second layer, the even-numbered qubits are swapped with the qubits to their right[2]. After n layers, all of the orbitals will have been adjacent to every other orbital exactly once and the ordering of the orbitals will have been reversed [66].

The hopping gates can then be folded into this swap network, since

$$\mathrm{FSWAP} \cdot \mathrm{Hopping}(\theta) = (S^\dagger \otimes S^\dagger) \cdot \mathrm{Hopping}(\theta + \pi/2), \tag{2.19}$$

---

[2] If n is odd, the last qubit in the first layer is left untouched. If n is even, the last qubit in the second layer is left untouched. Note that in the second layer, the first qubit is always left untouched.

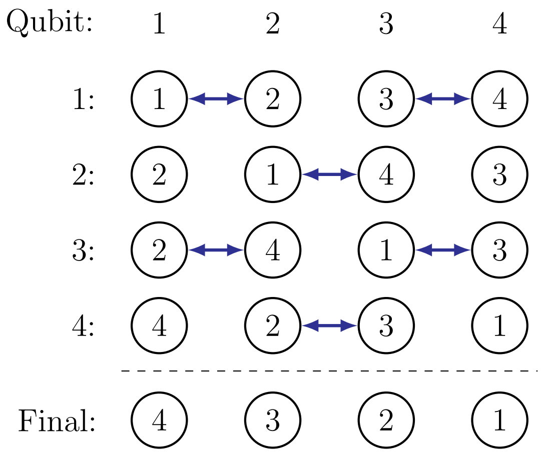


Figure 2.3: Generic swap network for four qubits done in four layers. The arrows represent FSWAP gates which are applied to qubits that are JW-adjacent. The JW ordering is a horizontal line from the first to the last qubit, with the numbers in the circles labelling the spin-orbitals.

where the S gate implements a $\pi/2$ phase shift. In total, n layers of two-qubit gates (plus some one-qubit corrections) are required to implement all possible hopping interactions between every pair of qubits.

The generic swap network provides us with an upper bound for the number of circuit layers required to implement the hopping gates. It is possible to improve upon this generic swap network by considering the pattern of hopping terms in the specific Hamiltonian that is being solved. For example, in Section 3.1.1 we describe an efficient swap network for the Hubbard model with $N_x \times N_y$ sites, which carries out all of the hopping terms in $2N_x$ layers of two-qubit gates.

2.4 Measuring the expectation value

At the end of each run of the ansatz circuit we must measure the energy of the ansatz state $|\psi(\boldsymbol{\theta})\rangle$ with respect to H, ideally in as few preparations of the circuit as possible. The most naïve method to achieve this would involve measuring $\langle H_i \rangle$ for each term H_i in H, then using

$$\langle H \rangle = \sum_i \langle H_i \rangle. \tag{2.20}$$

This method is inefficient, especially if there are a lot of terms in the Hamiltonian or many qubits. Instead, we can group together commuting terms and measure them all in parallel. There have been a number of recent works on general techniques for splitting the terms of a local Hamiltonian into commuting sets [101–106]; see [9] for a review. In the rest of this section we discuss which terms in H_F can be measured in parallel, and how to do the measurements.

All of the onsite and number operator terms can be measured simultaneously by doing a computational basis measurement on every qubit. The number terms are measured as

$$\langle n_i \rangle = \langle |1\rangle\langle 1|_i \rangle = \text{Prob(measure a 1 on qubit } i\text{)}. \tag{2.21}$$

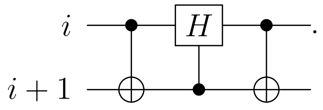
The onsite terms are measured as

$$\langle n_i n_j \rangle = \langle |11\rangle\langle 11|_{ij} \rangle = \text{Prob(measure a 1 on both qubit } i \text{ and } j\text{)}. \tag{2.22}$$

Calculating the expectation values of all the hopping terms $\langle a_i^\dagger a_j + a_j^\dagger a_i \rangle = \frac{1}{2}\langle X_i X_j + Y_i Y_j \rangle$ is more complex and requires multiple circuit preparations. Let us first focus on a single hopping term on JW-adjacent qubits i.e. $j = i + 1$. We need to measure the expectation $\frac{1}{2}\langle XX + YY \rangle$ on qubits i and $i+1$. One of the simplest ways of doing this is to first measure $\langle XX \rangle$[3], followed by $\langle YY \rangle$ at the next circuit preparation.

In fact, it is possible to measure the hopping term in just one circuit preparation by diagonalising $\frac{1}{2}(XX + YY)$. This comes at a cost of applying a two-qubit gate between qubits i and $i + 1$. A unitary M which diagonalises the hopping term is given by

$$M = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{2.23}$$



The result of the diagonalisation is

$$D = |01\rangle\langle 01| - |10\rangle\langle 10|^{4}, \tag{2.24}$$

so once computational basis measurements are done, the hopping term expectation value can be obtained via

$$\begin{aligned} \langle a_i^\dagger a_{i+1} + a_{i+1}^\dagger a_i \rangle &= \text{Prob(measure 1 on qubit } i \text{ and 0 on qubit } i+1) \\ &\quad - \text{Prob(measure 0 on qubit } i \text{ and 1 on qubit } i+1). \end{aligned} \tag{2.25}$$

---

[3] This can be done by applying the Hadamard gate to qubits i and $i + 1$, then doing a computational basis measurement.

[4] Using little-endian representation to index the computational basis states, see Section 6.1.

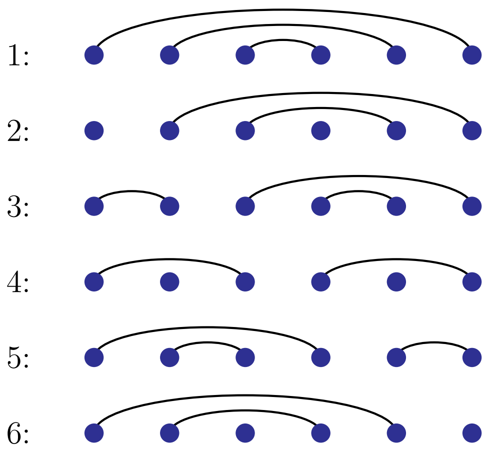


Figure 2.4: Measurement of all hopping terms on six qubits in six rounds via rainbows of non-crossing pairs. Each row corresponds to a measurement round and each pair of qubits is included in exactly one round.

This can be extended fairly simply to hopping terms $\frac{1}{2}(X_iX_j + Y_iY_j)Z_{i+1}\cdots Z_{j-1}$ between arbitrary qubits i and j. First, M is applied to qubits i and j, and equation (2.25) is used to work out the expectation $\frac{1}{2}\langle XX + YY\rangle$ (replacing $i+1$ with j and taking $i < j$ without loss of generality). The Z-strings are then dealt with by multiplying this expectation by a parity term. Doing a computational basis measurement on qubits $i+1$ through $j-1$ and counting the number of times that '1' is measured gives the parity term. If there are an even number of ones, the parity is 1, otherwise it is -1.

Due to the application of M, two different hopping terms that have the qubit i or j in common cannot be measured at the same time. However, M has the property that

$$M^\dagger(Z\otimes Z)M = Z\otimes Z, \tag{2.26}$$

allowing for the simultaneous measurement of sets of hopping terms under certain circumstances. Suppose we wish to measure the hopping term on the pair (i,j), where $i<j$ without loss of generality. We can simultaneously measure the term on (a,b) if $i<a,b<j$, or $a,b<i$, or $j<a,b$. In this case, we say that the pairs (i,j) and (a,b) are "non-crossing"; Figure 2.4 demonstrates why we have chosen this name. Since M preserves $Z\otimes Z$, statistics for measuring D and the Z-strings can be collected at the same time for non-crossing pairs. However, M does not preserve $Z\otimes I$; for example, if we had $i<a<j<b$, the statistics for the two pairs could not be collected simultaneously.

We can use this fact to measure hopping terms between every pair of qubits for an arbitrary number of qubits n using n preparations of the circuit. The idea is to produce a sequence of measurement rounds where each round contains at most two "rainbows". A rainbow between qubits i and j consists of the pairs

$$(i,j),(i+1,j-1),\dots,((i+j-1)/2,(i+j+1)/2) \tag{2.27}$$

when $j-i$ is odd, and the pairs

$$(i,j),(i+1,j-1),\dots,((i+j)/2-1,(i+j)/2+1) \tag{2.28}$$

when $j-i$ is even, corresponding to all non-crossing pairs between i and j centred at $(i+j)/2$. Then the kth round contains rainbows between qubits 1 and $k-1$, and between qubits k and n (where we do not include rainbows which have one end below qubit 1 or above qubit n). Each pair of qubits is then included in exactly one rainbow centred at their midpoint. This is illustrated for $n=6$ in Figure 2.4.

Similarly to the swap networks, it is possible to improve upon this generic measurement scheme by considering the pattern of hopping terms in the specific Hamiltonian being solved. For example, in Section 3.1.2 we describe how all the hopping terms in the Hubbard model can be measured in four circuit preparations.

We conclude this section with a brief discussion of a simple error detection procedure this measurement scheme allows. If the Hamiltonian to be solved preserves the number of electrons (or equivalently, Hamming weight after the JW transform), and the ansatz circuit for the VQE algorithm is also number preserving, then at every circuit preparation we can measure the Hamming weight without any additional cost. This is because the operator M is number preserving[5]. If the final measured state has a different Hamming weight to the starting state, then we can be confident an error has occurred.

2.5 Classical optimiser

The final part of the VQE algorithm is the classical optimisation routine used to choose the values of $\boldsymbol{\theta}$. The VQE algorithm makes many calls to the quantum computer to produce trial quantum states $|\psi(\boldsymbol{\theta})\rangle$. First we will lay out some of the terms that will be important in our analysis.

- Circuit evaluation/preparation = one run of the quantum computer.

---

[5] The Hadamard gate is not number-preserving. If we were measuring $\langle XX\rangle$, then we could not measure the Hamming weight simultaneously.

- Energy measurement $= x$ circuit evaluations. One measurement of all of the terms in H is obtained. For H_F, we showed in Section 2.4 that $x \leq n+1$ when there are n qubits.
- Energy estimate $= m$ energy measurements (also referred to as *function evaluation* in the context of optimisation routines).

The optimisation problem is stochastic since the function that we are optimising, $f(\boldsymbol{\theta}) = \langle\psi(\boldsymbol{\theta})|H|\psi(\boldsymbol{\theta})\rangle$, is noisy. We require an optimisation routine that is robust to the statistical noise from the measurements, and noise from the quantum computer (e.g. depolarisation, decoherence). Circuits to calculate analytic gradients can often not be implemented on NISQ devices due to the increased gate depth required. The optimisation routine will either have to use finite-difference methods to calculate gradients, or a gradient-free method.

Another property desirable in the optimiser is the ability for it to converge using a relatively low number of function evaluations. We can determine a rough budget for a reasonable number of calls to the quantum computer as follows. We start by assuming that we can perform each two-qubit quantum gate in 100ns and that measurements are instantaneous [107, 108]. If we assume for simplicity that the depth of the whole circuit is 100, and that the cost of classical computation is negligible, one circuit evaluation will take 10μs. If we want an accuracy of $\sim 10^{-2}$ in the energy estimate, 10^4 energy measurements (or $10^4 x$ circuit evaluations) are required (using that the error is $\sim 1/\sqrt{m}$ for m measurements). Using the quantum circuit time above, these $10^4 x$ circuit evaluations will take $0.1x$ seconds. Taking a representative number of 10^5 VQE loop iterations/function evaluations, this would take $10^4 x$ seconds $\approx 2.7x$ hours on the quantum computer.

There have been a large number of works recently on determining the best optimisation algorithms for VQE. For example, see [8, 9] for reviews and [109, 110] for detailed comparisons of various optimisation routines. In the rest of this section we will explain the details of two optimisation algorithms that we will be making use of in this thesis. One is a gradient-based optimiser called simultaneous perturbation stochastic approximation (SPSA), and the other is a gradient-free optimiser called coordinate descent (CD). Aside from gradient-based and -free methods, there has also been work done on machine learning and evolutionary strategy based optimisation methods [111, 112]. Research has also been conducted into measurement-frugal optimisers which aim to minimise the number of circuit evaluations. These have been shown to be competitive with, and in certain cases outperform, SPSA and

CD [113, 114]. Detailed descriptions of machine learning and measurement-frugal optimisation routines are outside the scope of this thesis.

2.5.1 Simultaneous perturbation stochastic approximation

One place to start looking for a suitable optimisation routine is the field of stochastic optimisation which deals with minimising/maximising functions which include random noise [115]. The SPSA algorithm [116] is one example of a stochastic optimisation algorithm. SPSA has been successfully used in small VQE experiments on superconducting hardware [35, 77, 117].

SPSA works in a similar way to the standard gradient descent algorithm, but instead of estimating the full gradient, a random direction is picked to estimate the gradient along. This is intended to make SPSA robust to noise and to require fewer function evaluations. Many aspects of this algorithm can be tailored to the specific problem at hand such as parameters that govern the rate of convergence, terminating tolerances, and the number of gradient evaluations to average the estimated gradient over.

Each gradient evaluation is estimated from two function evaluations (as compared with typically twice the number of parameters for finite difference methods) and is given by

$$g(\boldsymbol{\theta}_k) = \frac{f(\boldsymbol{\theta}_k + c_k \boldsymbol{\Delta}_k) - f(\boldsymbol{\theta}_k - c_k \boldsymbol{\Delta}_k)}{2c_k} \boldsymbol{\Delta}_k^{-1}, \tag{2.29}$$

where $\boldsymbol{\theta}_k$ is the current parameter vector after the kth step and c_k is the gradient step size. Each element in the vector $\boldsymbol{\Delta}_k$ is ± 1, chosen randomly with an equal probability for each outcome. The parameters are then updated via

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - a_k g(\boldsymbol{\theta}_k), \tag{2.30}$$

where a_k dictates the speed of convergence. Before carrying out the parameter update, it is possible to take several estimates of $g(\boldsymbol{\theta}_k)$ in random directions $\boldsymbol{\Delta}_k$ and use them to produce an averaged $g(\boldsymbol{\theta}_k)$.

The parameter and gradient step sizes are given by

$$a_k = \frac{a}{(k+1+A)^{\alpha}}, \quad c_k = \frac{c}{(k+1)^{\gamma}}, \tag{2.31}$$

where α, γ, a, A and c are the SPSA metaparameters. These metaparameters can be tailored to the specific problem that is being solved. Below we state what each of these parameters are and some guidelines for selecting values [118].

- α is the scaling exponent for the step size. $\alpha = 0.602$ is often recommended as a theoretically valid and practically effective value. $\alpha = 1$ is the asymptotically optimal value and may be used where there is a large amount of data.

- γ is the scaling exponent for the gradient step size. $\gamma = 0.101$ is often recommended as a theoretically valid and practically effective value. $\gamma = \frac{1}{6}$ is the asymptotically optimal value.

- a is the scaling parameter for the step size and is often chosen jointly with A. One recommended procedure for picking a is to choose it so that a_0 times the magnitude of elements in $g(\boldsymbol{\theta}_0)$ is approximately equal to the smallest change of magnitude desired in the elements of $\boldsymbol{\theta}$.

- A is the stability constant for the step size. It stops the algorithm from making very large jumps in the early iterations and becoming unstable. It is typically chosen to be around 10% (or less) of the maximum number of iterations.

- c is the scaling parameter for the gradient step size. A recommended value is approximately the standard deviation of the noise in $f(\boldsymbol{\theta})$.

In Sections 3.2.2 and 5.2.2 we discuss choices for these metaparameters, informed by numerical simulations, for our specific VQE optimisation problems. In Section 3.2.2, we also introduce a three-stage version of the SPSA algorithm, where different numbers of energy measurements are made at each stage of the algorithm.

2.5.2 Coordinate descent algorithm

We now describe an alternative algorithm, based on an approach independently discovered in [119–121]. The basic algorithm presented in these works can be applied to variational ansätze where the gates are of the form $e^{i\theta A}$ for Hamiltonians A such that $A^2 = I$. It is based on the observation that, for gates of this form, the energy of the corresponding output state is a trigonometric polynomial in θ (if all other variational parameters are fixed). This implies that it is sufficient to evaluate the energy at a small number of choices for θ in order to analytically determine its minimum with respect to θ. The algorithm proceeds by choosing parameters in some order (e.g. a simple cyclic sequential order, or randomly) and minimising with respect to each parameter in turn. It is shown in [119–121] that this approach can be very effective for small VQE instances.

We use a generalisation of this approach which works for any Hamiltonian with integer eigenvalues. This enables us to apply the algorithm to the HV ansatz because each gate in the ansatz can be seen as combining the gates of the form $e^{i\theta(XX+YY)/2}$, $e^{i\theta|11\rangle\langle 11|}$, and $e^{i\theta|1\rangle\langle 1|}$ (see Section 2.3.1). The corresponding Hamiltonians have eigenvalues $\{0,\pm 1\}$, $\{0,1\}$ and $\{0,1\}$ respectively. The generalisation is effectively the same as the one presented in [119, 121] to optimise over separate gates which share the same parameters.

The algorithmic approach has been given different names in the literature ("sequential minimal optimisation" [119], "Rotosolve" [120], "Jacobi diagonalisation" [121]). Here we prefer another name, coordinate descent (CD) [122], because this encompasses the approach we consider, whereas the above names technically refer to special cases of the approach which are not directly relevant to the algorithm we use[6].

Let A be a Hermitian matrix with eigenvalues $\lambda_k \in \mathbb{Z}$, and assume that $e^{i\theta A}$ is one of the gates (parametrised by θ) in a variational ansatz. Then the energy of the output state with respect to H can be written as

$$F(\theta) = \mathrm{Tr}[HUe^{i\theta A}|\psi\rangle\langle\psi|e^{-i\theta A}U^\dagger] \tag{2.32}$$

for some state $|\psi\rangle$ and unitary operator U that do not depend on θ. Writing $A = \sum_k \lambda_k P_k$ for some orthogonal projectors P_k and using linearity of the trace, this expands to

$$F(\theta) = \sum_{k,l} e^{i\theta(\lambda_k-\lambda_l)}\mathrm{Tr}[HUP_k|\psi\rangle\langle\psi|P_l U^\dagger]. \tag{2.33}$$

If Δ denotes the set of possible differences $\lambda_k - \lambda_l$, and $D = \max_{k,l}|\lambda_k - \lambda_l|$, this expression can be rewritten as

$$F(\theta) = \sum_{\delta\in\Delta} c_\delta e^{i\theta\delta} = \sum_{\delta=-D}^{D} c_\delta e^{i\theta\delta} \tag{2.34}$$

for some coefficients $c_\delta \in \mathbb{C}$. This is a complex trigonometric polynomial in θ of degree D. Therefore, it can be determined completely by evaluating it at $2D+1$ points. A particularly elegant choice for these is $\theta \in \{h\delta : -D \leq \delta \leq D\}$, where $h = 2\pi/(2D+1)$. Then the coefficients c_δ can be determined via the discrete Fourier transform:

$$c_\delta = \frac{1}{2D+1}\sum_{l=-D}^{D} e^{-ihl} f(hl). \tag{2.35}$$

---

[6] Sometimes the term "coordinate descent" is used for algorithms that perform gradient descent in each coordinate; we stress that here we *exactly* minimise over each coordinate.

To minimise F, we start by computing the derivative

$$\frac{dF}{d\theta} = i \sum_{\delta=-D}^{D} \delta c_\delta e^{i\delta\theta}, \tag{2.36}$$

and finding the roots of this function. To find these roots, we consider the function

$$G(\theta) = e^{2iD\theta}\frac{dF}{d\theta}. \tag{2.37}$$

Every root of $\frac{dF}{d\theta}$ is a root of $G(\theta)$, and as $G(\theta)$ is a polynomial of degree $2D$ in $e^{i\theta}$, its roots can be determined efficiently by computing the eigenvalues of the companion matrix of G. Finally, we ignore all roots that do not have modulus 1 (i.e. consider only roots of the form $e^{i\theta}$) and choose the root $e^{i\theta_{\min}}$ at which $F(\theta_{\text{min}})$ is smallest. Note that the only steps throughout this algorithm which require evaluation of $F(\theta)$ using the quantum hardware are the $2D + 1$ evaluations required for polynomial interpolation.

This argument also extends to the situation where we have m Hamiltonian evolution operations in the circuit that all depend on the same parameter θ; in this case, we obtain a trigonometric polynomial of degree mD (see [119, 121] for a proof), which is determined by its values at $2mD + 1$ points.

The above procedure demonstrated how to find the minimal parameter for a single θ (using $2D + 1$ function evaluations) whilst keeping the other variational parameters fixed. To carry out the full CD algorithm, this procedure is repeated for each parameter in turn, and then repeated again until a termination criteria is reached. This criteria could limit the number of passes through the parameters, or it could halt if the θ's stop changing – indicating that a minima has been reached.

Chapter 3

Solving the Hubbard model using the variational quantum eigensolver

There are a number of important questions which must be answered to understand whether solving the Hubbard model is a realistic target for near-term quantum computers. These include:

- Which ansätze can effectively parametrise the ground state?
- What is the precise complexity of implementing the variational ansatz and of carrying out the measurements?
- How well will the optimisation routines used handle statistical noise, and noise in the quantum circuit?

In this chapter we address these questions in order to estimate how well realistic near-term quantum computers will be able to solve the Hubbard model. We develop detailed resource estimates and circuit optimisations, as well as carrying out extensive numerical experiments for grids with up to 12 sites (24 qubits). Although the Hubbard model is easily solvable directly by a classical algorithm for systems of this size, these experiments give insight into the likely performance of VQE on instances that are beyond the capability of classical hardware. In the NISQ regime, it is essential to carry out precise complexity calculations to understand the feasibility of the VQE approach.

A number of works have applied VQE to the Hubbard model. Wecker et al. [92] developed the HV ansatz, which is a key tool that we use and expand upon. They tested it for the half-filled Hubbard model for systems of up to 12 sites – in the case of simulated exact energy estimates, they used ladders with dimensions $N_x \times 2$ for $N_x = 2, \ldots, 6$; in the case of realistic energy estimates, they tested a system of size 4×2. Implementation of two layers of this ansatz for a 4×2 system would require around 1000 gates according to their estimate (we reduce this estimate substantially; see Section 3.1.1).

Reiner et al. [93] have studied how gate errors affect the HV ansatz. They considered a model where gates are subject to fixed unitary over-rotation errors, and found that for small system sizes (grids of size 2×2, 3×2 and 3×3), reasonably small errors did not prevent the variational algorithm from finding a high-quality solution. Verdon et al. [111] developed an approach to optimising VQE parameters using recurrent neural networks, and applied it to Hubbard model instances of size 2×2, 3×2 and 4×2. Dallaire-Demers et al. [87] developed a low-depth circuit ansatz inspired by the UCC ansatz and applied it to the 2×2 Hubbard model.

Concurrently with this work, Cai [94] determined detailed theoretical resource estimates for applying the HV ansatz to solve the Hubbard model, using silicon spin qubits as the quantum hardware. The circuit complexities obtained are qualitatively similar to ours but are not directly comparable since they use a more restrictive gate set and topology aimed at efficient implementation on a specific hardware platform. More recently, small-scale experiments on quantum hardware using up to four qubits have been demonstrated for solving the Hubbard model using VQE [117, 123].

We start this chapter with theoretical work describing how we can efficiently implement the HV ansatz (and a generalisation we call the number-preserving (NPr) ansatz) and how to measure the expectation values on a quantum computer. We follow this with extensive numerical simulations with three levels of realism: assuming we can directly extract the exact value of $\langle\psi(\boldsymbol{\theta})|H|\psi(\boldsymbol{\theta})\rangle$; simulating measurements to represent an ideal quantum computer; and simulating the effect of noise during the circuit. The work in this chapter is based on Sections I-III and Appendices B and D of the paper "Strategies for solving the Fermi-Hubbard model on near-term quantum computers" [57].

3.1 VQE implementation details

We are concerned with finding the ground state of the Hubbard model

$$H_{\text{hub}} = -t \sum_{\langle i,j\rangle,\sigma} (a_{i\sigma}^\dagger a_{j\sigma} + a_{j\sigma}^\dagger a_{i\sigma}) + U \sum_i n_{i\uparrow} n_{i\downarrow} \tag{3.1}$$

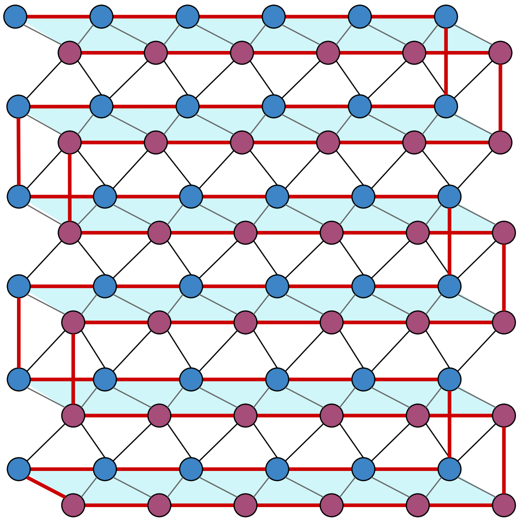


Figure 3.1: An illustration of how fermionic modes can be mapped to physical qubits on a physical architecture such as Google's Sycamore device [1]. The fermionic modes (blue: spin-up, red: spin-down) on a 6×6 grid are mapped to qubits in an array of size $2 \times 6 \times 6$. The red line represents the order associated with the JW encoding of the qubits, which moves from the top left towards the right. The blue panels are added to aid visualisation. Note that the red line does not follow the true connectivity of the qubits (the thin black lines), and hence any 'local' operator with respect to the JW encoding is not necessarily local with respect to the physical connectivity of the qubits, and vice versa.

on an $N = N_x \times N_y$ grid occupied by N_{occ} fermions, with open boundary conditions. The notation $\langle i, j \rangle$ indicates that the sum is performed over nearest-neighbour (horizontal and vertical) sites i, j in the grid, this is the simplest version of the Hubbard model that was introduced in Section 1.3.

The first step in the VQE algorithm is to apply a fermion-to-qubit encoding to this Hamiltonian. We use the JW encoding (see Section 2.2) which maps one spin orbital to one qubit, i.e. for an N site Hubbard model, we require $2N$ qubits. Although this is one of the simplest encodings, for small grid sizes where $\min(N_x, N_y) \leq 8$ it appears to be the most efficient approach in terms of qubit count and circuit depth required [57].

The JW encoding maps the fermionic modes to a line, therefore an ordering must be chosen. We use a"snake"-shaped configuration [45, 94, 100], which is demonstrated in Figure 3.1 superimposed on a device with Google Sycamore architecture [1]. All of the spin-up electrons are followed by all of the spin-down, and within each spin-type, the sites take on the snake ordering. The advantage of using this configuration is that in Section 3.1.1 we can make use of fermionic swap

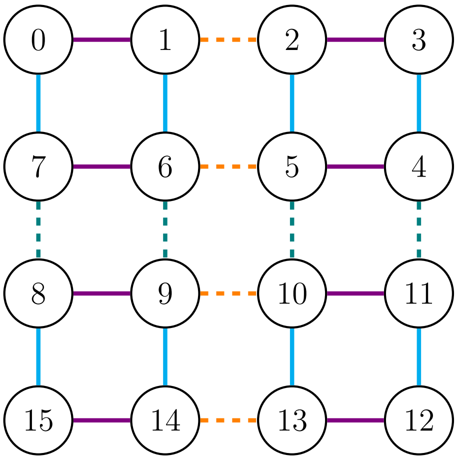


Figure 3.2: The four sets of hopping terms (for a fixed spin). Hopping terms of the same colour commute, and hence in principle can be implemented simultaneously. Purple corresponds to the horizontal terms h_1, dashed orange to h_2, blue to the vertical terms v_1 and dashed green to v_2.

networks for efficiently implementing the ansatz circuits, and carry out Hamiltonian measurements using the lowest number of circuit preparations.

The next step in the VQE algorithm is choosing an appropriate ansatz circuit. We will be making use of the HV ansatz which was discussed in Section 2.3.1. Recall that in the HV ansatz we split up a Hamiltonian into the sum $H = \sum_j H_j$ where the terms inside H_j are commuting, and then apply evolutions of the form $e^{i\theta H_j}$ to the starting state. In the case of the Hubbard model, one layer of the HV ansatz is a unitary operator of the form

$$e^{i\theta_{v_2} H_{v_2}} e^{i\theta_{h_2} H_{h_2}} e^{i\theta_{v_1} H_{v_1}} e^{i\theta_{h_1} H_{h_1}} e^{i\theta_o H_o}, \tag{3.2}$$

where H_o is the onsite term; H_{v_1} and H_{v_2} are the vertical hopping terms; H_{h_1} and H_{h_2} are the horizontal hopping terms as shown in Figure 3.2. Different layers can have different parameters. Note that there is some freedom in the order with which we can implement these terms. In addition, some of terms may not be needed depending on the grid dimensions.

The HV ansatz has been shown to be effective for small Hubbard model instances [92, 93], and involves a small number of variational parameters: at most 5 per layer. One disadvantage of this ansatz is that preparing the initial state $|\psi_0\rangle$ (the ground state of the non-interacting $U = 0$ Hubbard model) is a nontrivial task. It can be produced using the 2D fermionic Fourier transform, for which efficient algorithms are known [100, 124], or via a direct method based on the use of Givens rotations [100]. For grids of size up to 20 × 20, using Givens rotations is the most efficient [57] method; on a fully-connected architecture it leads to a circuit depth of $N-1$.

The number-preserving ansatz

It is possible to avoid the depth overhead for constructing the non-interacting ground state if an ansatz is used that allows for simpler initial states (such as computational basis states). Here we introduce one such ansatz, the NPr ansatz, which is a generalisation of the HV ansatz. A trade-off for being able to use simpler initial states is that the NPr ansatz uses more parameters, making the optimisation process more challenging.

The NPr ansatz is derived from the HV ansatz by replacing all hopping and onsite terms with a more general number-preserving operator[1] parametrised by two angles θ and ϕ, and implemented by the two-qubit unitary

$$U_{\mathrm{NPr}}(\theta,\phi)=\begin{pmatrix}1 & 0 & 0 & 0\\ 0 & \cos\theta & i\sin\theta & 0\\ 0 & i\sin\theta & \cos\theta & 0\\ 0 & 0 & 0 & e^{i\phi}\end{pmatrix}=e^{i\theta(XX+YY)/2}e^{i\phi|11\rangle\langle 11|}. \tag{3.3}$$

The non-interacting ground state can still be used as the initial state, although computational basis states (where the Hamming weight is equal to the fermionic occupation number of interest) can also be used with some success (see Appendix A). One layer of the ansatz then consists of applying a $U_{\mathrm{NPr}}(\theta,\phi)$ gate across each pair of qubits that correspond to fermionic modes that interact according to the Hubbard Hamiltonian H_{hub}, i.e. where there are hopping or onsite interactions. As before, different layers can have different parameters.

For an $N=N_x\times N_y$ grid, one layer of the NPr ansatz requires

$$2\left(2\big[N_x(N_y-1)+N_y(N_x-1)\big]+N_xN_y\right)=10N-4N_x-4N_y \tag{3.4}$$

parameters. The HV ansatz is the special case of the NPr ansatz that also preserves spin and where many parameters are fixed to be identical or zero. Although the NPr ansatz has more parameters, the gate depths for the two ansätze will be the same as they both require the application of gates where there are hopping and onsite terms.

---

[1] This is similar to the exchange-type entangling gates discussed in [77, 81]; an alternative notion of number-preserving VQE ansatz was studied in [89].

Architecture	Ansatz circuit depth per layer				
	N_x even	N_x odd	4×4	5×5	6×6
Fully-connected	$2N_x + 1$	$2N_x + 2$	9	12	13
Nearest-neighbour	$4N_x - 1$	$4N_x$	16	21	24
Google Sycamore	$6N_x + 1$	$6N_x + 2$	25	32	37

Table 3.1: Circuit depths per layer of the EHV and NPr ansätze for various architectures.

In Section 3.1.1 we will explain how to implement the HV and NPr ansätze efficiently on a quantum computer with arbitrary qubit connectivity. We will then discuss how the next step of the VQE algorithm, the measurement of the expectation value $\langle \psi(\boldsymbol{\theta}) | H_{\text{hub}} | \psi(\boldsymbol{\theta}) \rangle$, can be done efficiently in Section 3.1.2. We will extend these in Section 3.1.3 to more realistic hardware architecture such as nearest-neighbour connectivity and the Google Sycamore device. The resulting two-qubit gate depths for these different architectures are summarised in Table 3.1. We leave the discussion of the final step of the VQE algorithm, the classical optimisation routine, to the numerical simulations in Section 3.2.2.

3.1.1 Efficient ansatz circuit implementation

A key ingredient in the complexity calculations for our circuits will be their depths. To compute this, we assume that the quantum computer can implement arbitrary two-qubit gates between connected qubits, and that one-qubit gates can be implemented at zero cost. These assumptions are not too unrealistic. Almost all the two-qubit gates we will need are rotations of the form $e^{i(\theta(XX+YY)+\gamma ZZ)}$ (up to one-qubit unitaries), which can be implemented natively on some superconducting qubit platforms; and one-qubit gates can be implemented at substantially lower cost (e.g. shorter gate times with a higher fidelity) in some architectures [107]. For now we will also assume that two-qubit gates can be applied across arbitrary pairs of qubits i.e. we have a fully-connected architecture.

We will now analyse how one layer of the HV ansatz (and hence also the NPr ansatz) from equation (3.2) can be carried out efficiently. All of the onsite gates $e^{i\theta_o H_o}$ can be implemented in depth one as they act on disjoint pairs of qubits. Taking Figure 3.1 as an example, doing the onsite gates corresponds to doing U_{NPr} between pairs of blue and red qubits that are connected by a thin black line on the blue panels. All of the horizontal hopping terms $e^{i\theta_{h_2} H_{h_2}}, e^{i\theta_{h_1} H_{h_1}}$ can be carried out in depth two, since these correspond to gates between qubits that are adjacent in the JW ordering. The vertical hopping terms $e^{i\theta_{v2}H_{v2}}, e^{i\theta_{v1}H_{v1}}$ are more complicated to implement since they are accompanied by strings of Z operators due to the JW encoding. For example, the vertical hopping term between qubits 0 and 7 in Figure 3.2 is an 8-qubit gate which can be costly to implement (see Figure 2.2). Therefore, we must make use of fermionic swap networks [66].

We will now restrict our analysis to one spin-type as the horizontal and vertical hopping terms act on qubits with the same spin. There are $N = N_x \times N_y$ qubits, therefore the generic swap network described in [66] and Section 2.3.2 would require N layers of two-qubit gates to implement all possible hopping terms. Here we present a scheme that implements all of the horizontal and vertical hopping terms in $2N_x$ layers for N_x even and $2N_x + 1$ for N_x odd. It is based on a technique that Kivlichan et al. [66] used to simulate Trotter steps of the Hubbard model. In particular, we remove some unnecessary vertical fermionic swap gates and instead only swap horizontally adjacent qubits. This means that, for an $n \times n$ grid, only n repetitions of a column-permuting sub-routine (which itself has depth two) are necessary to be able to implement all vertical hopping terms locally, in comparison to the $\frac{3}{\sqrt{2}}n$ iterations that are deemed to be necessary in [66]. In what follows, when we say that an operator is implemented locally, we mean that the two qubits that it acts on are JW-adjacent.

We repeatedly apply the operator $U_R U_L$, where U_L swaps odd-numbered columns with those to their right, and U_R swaps even-numbered columns with those to their right. After each application of $U_R U_L$, a new set of qubits that were previously not vertically JW-adjacent are made JW-adjacent, meaning that the vertical hopping interaction between them can be implemented locally using a single number-preserving operator, without Z-strings. For an $N_x \times N_y$ grid, it suffices to apply $U_R U_L$ a total of N_x times (giving a two-qubit gate depth of $2N_x$) to allow all vertical interactions to be implemented locally and return the qubits to their original positions. This is demonstrated in Figure 3.3 for a 4×4 grid.

Note that the vertical terms are implemented in a different order to the horizontal terms. If the columns begin in the order $1, 2, 3, 4 \ldots, N_x$ (assuming that N_x is even), then after a single application of $U_R U_L$, they are re-ordered to $2, 4, 1, 6, \ldots, N_x - 3, N_x - 1$. Each subsequent application of $U_R U_L$ will place a new even-numbered column to the far-left, and a new odd-numbered column to the far-right, so that after $N_x/2$ applications each column will have been at the far-left or -right exactly once. Since it is at the far ends that vertical terms can be applied locally, then after $N_x/2$

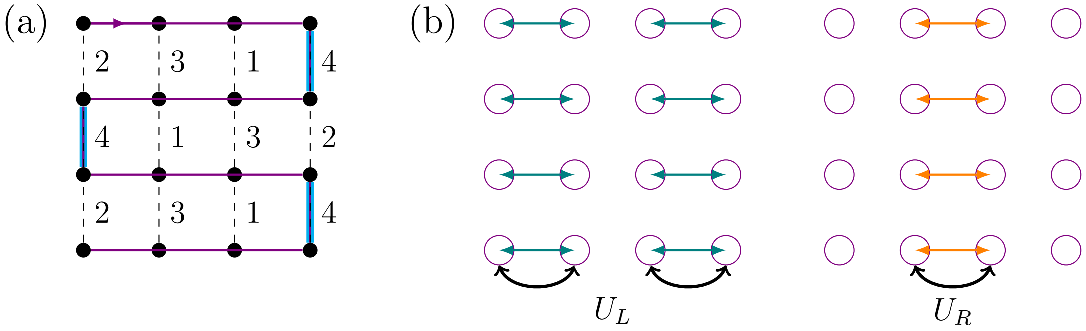


Figure 3.3: (a) Vertical hopping term implementation for a 4×4 Hubbard model grid. The numbers i show which vertical term will be implemented after i applications of $U_R U_L$. The highlighted blue lines show the only places where the hopping terms can be implemented – at the JW-adjacent positions. (b) Action of U_L and U_R on the grid of qubits.

applications of $U_R U_L$ (a depth N_x circuit), all terms that can be applied locally at the left will have been applied for the even-numbered columns, and similarly for the odd-numbered columns. Applying another $N_x/2$ iterations of $U_R U_L$ will move all the even-numbered columns move to the far-right, and all odd-numbered columns to the far-left, which allows the remaining terms to be implemented locally.

This circuit can be used to implement all vertical hopping terms in depth $2N_x$ for even N_x, and depth $2N_x+1$ for odd N_x. This is because for even N_x the hopping terms can be implemented in parallel with U_R, and for odd N_x some hopping terms can be implemented in parallel with U_L and others with U_R; one hopping term is left over in the latter case, leading to an overall overhead of depth one. The horizontal hopping terms can be folded into this swap network using equation (2.19). The onsite terms cannot be folded in, hence the final depth of the circuit that implements one layer of the ansatz is $2N_x + 1$ for N_x even and $2N_x + 2$ for N_x odd. Figure 3.4 shows the circuit used to implement a single layer of the ansatz for a 4×4 grid, for just one of the spins (and therefore omitting the onsite interactions).

We stress that this efficient version of the HV ansatz is different from the standard HV ansatz in equation (3.2), in that vertical hopping terms are implemented in a different order. We refer to it as the efficient HV (EHV) ansatz for the remainder of this chapter. We also remark that all this discussion has assumed the use of open boundary conditions in the Hubbard model. Periodic boundary conditions in the horizontal direction can be implemented without any overhead, but periodic boundaries in the vertical direction are significantly more challenging. However, smooth boundary conditions, which can be even more advantageous in terms of reducing

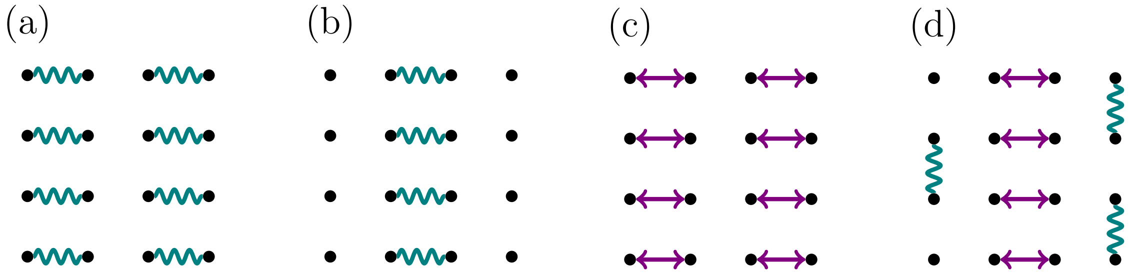


Figure 3.4: Gates required to implement one layer of the EHV or NPr ansatz for a single spin-type. Circuit layers go from (a) to (d), with (c) and (d) repeated three more times to complete the swap network. Wavy green lines are U_{NPr} and purple arrows are FSWAP gates. (c) represents U_L and (d) represents U_R implemented in parallel with vertical hopping terms. In our implementation, (a) is combined with the first (c), and (b) with the last (d), allowing horizontal hopping terms to be combined with the FSWAP gates.

finite-size effects [125], can also be implemented without an overhead.

Finally, we give an estimate of the number of two-qubit gates required for a complete run of the whole circuit for EHV or NPr ansatz. The cost of preparing the initial state is at most $2(N-1)\lfloor N/2 \rfloor$ gates, where $N = N_x N_y$. The cost of the ansatz circuit itself is at most the depth per layer multiplied by the maximal number of two-qubit gates applied per step of the circuit (which is at most N), multiplied by the number of layers. Finally, there is a cost of at most N for the two-qubit gates required for performing the final measurement. For example, in the case of a fully-connected architecture, the gate count for a circuit with L layers is at most

$$(N-1)N + (2N_x + 1)NL + N \tag{3.5}$$

for even N_x, and

$$2(N-1)\lfloor N/2 \rfloor + (2N_x + 2)NL + N \tag{3.6}$$

for odd N_x. In the special case of a 2×4 system with two layers, using a more careful calculation, we obtain a bound of at most 36 gates per layer, giving an upper bound of 136 gates in total. By contrast, the estimate for this case in [92] was around 1000 gates, more than a factor of 7 higher.

3.1.2 Measurement scheme

After each run of the ansatz circuit, we need to measure the energy of the state produced with respect to H_{hub}. For an $N = N_x \times N_y$ grid, there are $4N - 2N_x - 2N_y$ hopping terms and N onsite terms that need to be measured. It turns out that all

Type	Alternating $\uparrow, \downarrow$		All $\uparrow$, then all $\downarrow$	
	Sequential	Snake	Sequential	Snake
Onsite	1	1	1	1
Horizontal hopping	3	3	2	2
Vertical hopping	$2N_x + 1$	2	$N_x + 1$	2
Total	$2N_x + 5$	6	$N_x + 4$	5

Table 3.2: Number of circuit preparations needed to measure all of the Hubbard model terms (using the non-crossing measurement method) for different JW orderings. Note that we have not included the possibility of rearranging the ordering after the ansatz circuit; for example, for the alternating snake ordering, we could measure all of the horizontal hopping terms in two circuit preparations using an additional layer of SWAP gates.

of these terms can be measured in five circuit preparations if we make use of the notion of non-crossing measurements introduced in Section 2.4.

All of the onsite terms can be measured in one circuit preparation by doing computational basis measurements on every qubit. If we break the hopping terms into four sets of commuting terms – two horizontal and two vertical as displayed in Figure 3.2 – then each of the groups can be measured using one circuit preparation. The hopping terms in each of these groups are non-crossing and so we can simultaneously apply the operator M in equation (2.23) to pairs of qubits corresponding to these terms (for example, for the first set of vertical hopping terms, these would be the pairs $(0,7), (1,6), \ldots, (11,12)$), and collect statistics as described in Section 2.4. Note that, in our scheme, measurement is the one point in the circuit where quantum gates need to be applied across qubits that are not JW-adjacent.

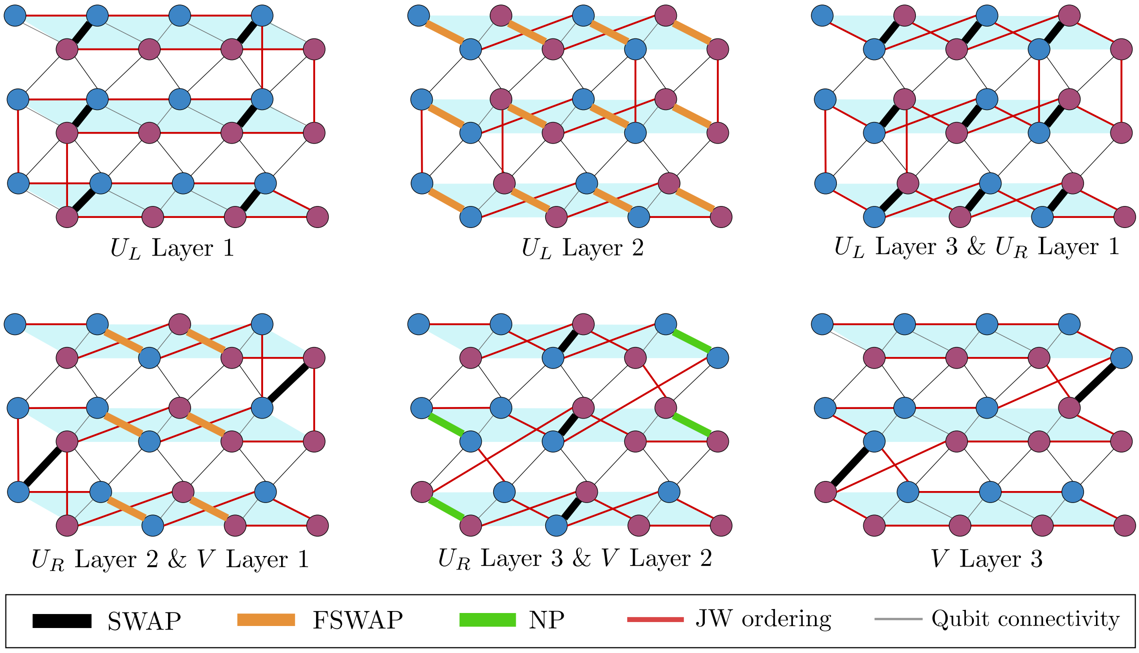
Recently, Cai [94] described an alternative approach to obtaining the expectation value using five measurements, based on switching the JW ordering around at different circuit preparations. The snake ordering is changed to go up and down, as opposed to left and right (see Figure 3.2) at circuit preparations where we wish to measure the vertical hopping terms. This makes the vertical hopping terms the JW-adjacent ones and hence removes the Z strings. The cost of implementing this approach would be similar to the approach proposed here in the case of square grids (or perhaps slightly more efficient). For non-square grids the approach proposed here will be more efficient, as one can choose the orientation of the grid to minimise the length of Z-strings for the ansatz circuit implementation.

We finish this section with a demonstration of how the JW ordering can change the number of circuit preparations required for measuring all of the terms in H_{hub}. In Table 3.2 we compare the snake ordering with the sequential ordering of sites (the columns are numbered from left to right for every row, i.e. the second row will read 4, 5, 6, 7 in Figure 3.2), and the case where we alternate spin-up and -down versus where each spin is grouped together. In the worst case scenario, an extra $2N_x$ circuit preparations are required to measure all of the terms in the Hamiltonian. This highlights the need for choosing an optimal JW ordering for performing measurements, but we stress that the choice of JW ordering must also be informed by the ansatz circuit.

3.1.3 Implementation on realistic hardware

The description of the EHV and NPr ansätze from Section 3.1.1 assumes that two-qubit gates can be implemented across arbitrary pairs of qubits. Most quantum computing architectures have restrictions on their connectivity. These architectures will in general require additional swap operators to move pairs of qubits into positions in which they can interact, and then to move them back again. However, almost all of the gates that are applied in the ansätze take place along the 1D line of the JW ordering; the only other gates are onsite terms. This means that these ansätze can be implemented on a $2 \times (N_x N_y)$ nearest-neighbour architecture with no additional gate depth per circuit layer. However, this approach would require an overhead scaling with N_x to measure the vertical hopping terms, e.g. qubits will need to be swapped to apply M between sites 0 and 7 in Figure 3.2. It would also require the qubit layout to be particularly long and thin (or a larger lattice of which this would be a subgraph). In this section we describe alternative approaches to implement the EHV and NPr ansätze on realistic architectures whose shape is closer to the shape of the grid itself.

Once we have decided on a qubit layout, we can consider the cost of implementing the operator $U_R U_L$ from Section 3.1.1, and how it can be combined with the vertical hopping terms. Since vertical hopping terms are always applied in the same positions (those pairs of qubits that are vertically JW-adjacent), the same operator, V, is used to apply all of them (one round at a time). The depth of the circuit required to implement one layer of the ansatz will then be determined by the depth of the circuit required to implement $V U_R U_L$, which is repeated N_x times, plus the depth of the circuits used to implement the horizontal hopping and onsite terms.


U_L Layer 1

U_L Layer 2

U_L Layer 3 & U_R Layer 1

U_R Layer 2 & V Layer 1

U_R Layer 3 & V Layer 2

V Layer 3

Figure 3.5: Implementation of the operator VU_RU_L (each split into 3 layers) on the Google Sycamore architecture for even N_x, shown here for a 4×3 grid. The red lines represent the ordering of qubits due to the JW encoding, and the thin black lines the connectivity of the qubits.

Nearest-neighbour connectivity

On a nearest-neighbour architecture, we could use a qubit layout similar to that described in Figure 3.1, but where the lattice consists of alternating rows of spin-up and spin-down qubits. In this layout, horizontally JW-adjacent qubits are physically adjacent, but vertically JW-adjacent qubits are not. This means that the operators U_L and U_R, which swap horizontally JW-adjacent qubits, can be implemented directly in depth one each. However, the operator V requires that each pair of vertically JW-adjacent qubits are moved so that they become physically adjacent, and then moved back again, which can be achieved using two layers of SWAP gates. The first layer of SWAP gates can be implemented in parallel with the U_R operator (for even N_x[2]), meaning that VU_RU_L can be implemented by a circuit of depth four. Also, as discussed in Section 3.1.1, we can fold the horizontal hopping interactions into the swap network. Finally, all onsite interactions can be implemented in depth one. This yields a final circuit depth of $4N_x + 1$ per layer for N_x even. This could be decreased to $4N_x$ – to match the approach taken by Cai [94] – by combining onsite interactions with swapping operations, although this would change the ordering of the interactions performed in the ansatz.

---

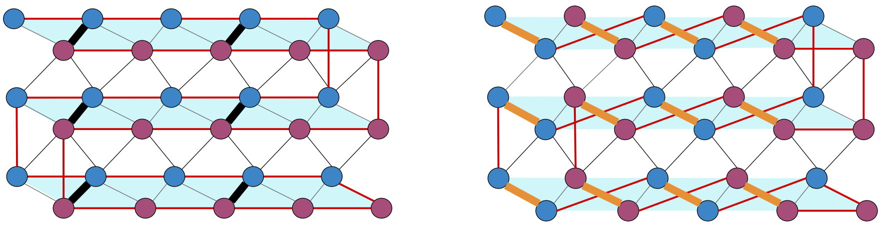
[2] For odd N_x, some of the SWAP gates can be implemented in parallel with U_R, and others with U_L. In the end this incurs an extra overhead of only depth one, using an approach similar to that described in Figure 3.6.

The above interlaced approach would result in a physical lattice of shape $N_x \times (2N_y)$. However, instead of alternating rows of spin-up and spin-down, we can also place the spin-up grid physically next to the spin-down grid. This results in a lattice of shape $(2N_x) \times (N_y)$. The horizontally and vertically JW-adjacent terms are then adjacent on the physical lattice as well, and we can carry out these terms as described in Section 3.1.1. However, the qubits between which we want to implement onsite terms are distance N_x from each other. Using a swap network of depth $N_x - 1$, where the ith layer swaps i pairs of adjacent qubits starting from the middle of each row, we can bring the required qubits next to each other. We then perform the onsite gate and use $N_x - 1$ more layers to swap to the original position. This approach gives a final depth of $4N_x - 1$ for even N_x, and $4N_x$ for odd N_x which is a slight improvement on the interlaced approach. Also, the interlaced approach requires an additional layer of SWAP gates at the end of the algorithm to measure vertical hopping terms, which is not required for the separated approach.

Google Sycamore architecture

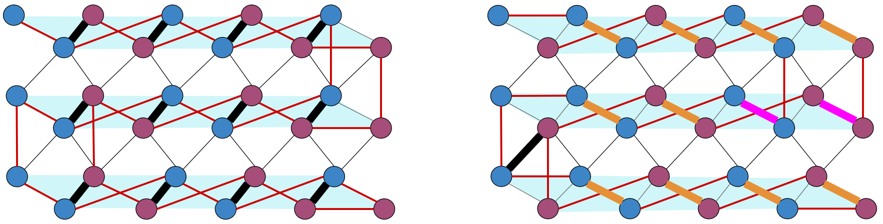
As another example, we consider how to implement the EHV and NPr ansätze efficiently on Google's Sycamore architecture [1]. Once again we are concerned with the depth of the circuit required to implement VU_RU_L. In the Sycamore architecture, no JW-adjacent qubits are physically adjacent and so each of U_R, U_L, and V must be split into 3 layers each: one to swap qubits into physically adjacent positions; one to carry out the required interaction; and one more to swap the qubits back to their original positions. Many of these layers overlap and can be implemented in parallel. Figure 3.5 illustrates how to implement the operator VU_RU_L with a circuit of depth 6 for even values of N_x.

As in Section 3.1.1, we can fold the horizontal hopping interactions into the swap network, and all onsite interactions can be implemented in depth one. This yields a final circuit depth of $6N_x + 1$ per layer for even values of N_x. For odd values of N_x, we lose the ability to implement the vertical hopping terms in parallel with the operator U_R, which increases the depth of the final circuit. In Figure 3.6 we show how to implement the operator VU_RU_L in depth 7. Here (but not in the even N_x case), the first and last layers can be implemented in parallel, and so we obtain a



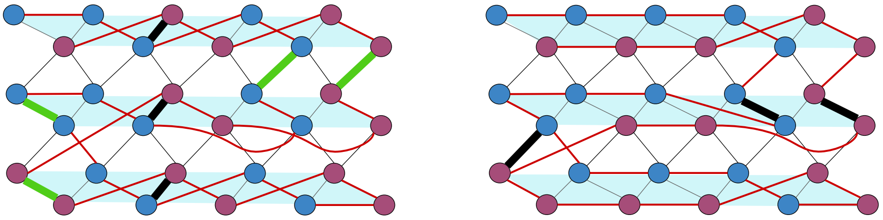
U_L Layer 1

U_L Layer 2



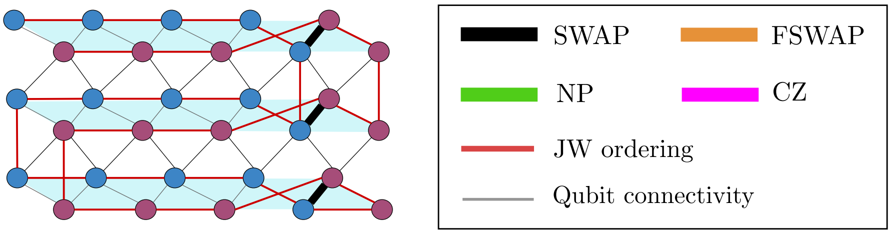
U_L Layer 3 & U_R Layer 1

U_R Layer 2 & V Layer 1



U_R Layer 3 & V Layer 2

V Layer 3



V Layer 4

Figure 3.6: Implementation of the operator VU_RU_L on the Google Sycamore architecture for odd N_x, shown here for a 5×3 grid. Note the controlled-Z (CZ) gates in the fourth layer of the circuit are a combination of the FSWAP gate from Layer 2 of U_R and the SWAP gate from Layer 1 of V.

final circuit depth for the ansatz of $6N_x + 2$ per layer, one more than in the even case.

3.2 Numerical results

When simulating a VQE experiment on a classical computer, we will consider three different levels of realism:

- The simplest but least realistic level is to assume that we can obtain exact energy estimates to learn $\langle\psi(\boldsymbol{\theta})|H_{\text{hub}}|\psi(\boldsymbol{\theta})\rangle$, which can be used directly as input to a classical optimiser. This will allow us to test which ansätze are effective, because an ansatz circuit which performs poorly at this level of realism will likely not be suitable for use on a quantum computer. This is covered in Section 3.2.1.
- The next level of realism is to simulate the result of measurements as if they were performed on a quantum computer, but to assume that the quantum computer is perfect, i.e. does not experience any noise. This level allows us to test how well the classical optimisers handle statistical effects due to random measurements and gives an idea of the best performance the VQE algorithm could have in practice. This is covered in Section 3.2.2.
- Finally, we can simulate the effect of noise during the quantum computation. For this we use a depolarising noise model. These simulations allow us to test how well the optimisers cope with additional noise and to test simple error detection procedures. This is covered in Section 3.2.3.

We developed a high-performance software tool in C++, based on the quantum simulator QuEST [56] which enabled the ansätze we used to be validated and compared. The tests were mainly carried out on the Google Cloud Platform. In the preliminary tests, we found that GPU-accelerated QuEST commonly outperformed QuEST running on CPU only (whether single-threaded, multi-threaded, or distributed). For most of the results reported here, we found a speed-up of 4-5× when compared with a 16 vCPU machine available on Google Cloud, which is similar to the speed-up reported in [56]. The GPU-accelerated tests were carried out using a single vCPU machine equipped with either Nvidia Tesla P4 or Nvidia Tesla K80. Some of the noisy experiments were carried out on a single vCPU instance, as for some of the smaller grid sizes it was found that a single CPU performs similarly to a GPU-accelerated version (for small grid sizes, the data transfer between CPU and GPU dominates the run-time).

For realistic energy measurements we obtained a significant speedup by storing the probability amplitudes of the final state produced by the circuit. Computational basis measurements on that state were then simulated by sampling from this distribution, hence avoiding the need to rerun the circuit; see Section 6.3 for more details. This optimisation is not available with noisy circuits, so those tests are much more computationally intensive.

Occupied orbitals	Grid sizes
2	1×2, 1×3, 2×2
3	1×4
4	1×5, 1×6, 2×3
6	1×7, 1×8, 2×4, 3×3
7	1×9
8	1×10, 1×11, 2×5, 2×6
9	1×12, 3×4

Table 3.3: Number of occupied orbitals corresponding to the lowest energy of the $t = 1, U = 2$ Hubbard Hamiltonian for each grid size tested.

We now outline some implementation decisions that were made. First, unless specified otherwise, we ran VQE for the $t = 1, U = 2$ Hubbard model and started with the number of occupied orbitals that corresponds to the lowest energy of the Hamiltonian H_{hub} (not, for example, the half-filled case as in [92]). These occupation numbers are listed in Table 3.3. The ansätze we use preserve fermion number, so remain in this subspace throughout the optimisation process.

In these simulations we ran the HV, EHV and NPr ansätze. For the HV ansatz, there is a flexibility in the ordering of Hamiltonian terms for time-evolution (see equation (3.2)). In the case of $1 \times N_y$ grids, we used a o, v_1, v_2 ordering. For $2 \times N_y$ grids, we used a o, h_1, v_1, v_2 ordering (except 2×2, where there is no v_2 term). For $3 \times N_y$ grids, we used an o, h_1, v_1, v_2, h_2 ordering. By contrast, for the two efficient ansätze, this ordering is largely pre-determined, except that we have a choice of when to implement the onsite terms in the EHV ansatz; we chose to do so at the start of each layer.

For all ansätze, one needs to choose initial parameters. We used a simple deterministic choice of initial parameters, which (similarly to [93]) were all set to $1/L$, where L is the number of layers. We also experimented with choosing initial parameters at random, we found that as long as the initial parameters were small, e.g. $[0, \pi/10]$, a similar performance was achieved. Unless otherwise specified, the initial state was the ground state of the non-interacting Hubbard model.

3.2.1 Exact simulations

We will now present the simulation results for the first level of realism which assumes we have direct access to the expectation value $\langle \psi(\boldsymbol{\theta}) | H_{\text{hub}} | \psi(\boldsymbol{\theta}) \rangle$. Unless stated otherwise, we use the implementation choices detailed above. For the classical optimisation routine we use the Limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) optimisation algorithm. We evaluated different optimisation methods given in the NLopt C library for nonlinear optimisation [126] and found that L-BFGS was usually a very effective algorithm to use when considering a perfect, noiseless, version of VQE with simulated exact energy estimates. Other algorithms required many more iterations, or often found lower-quality local minima. To estimate the gradient, as required for L-BFGS, we used a simple finite difference approximation.

Ability to represent the ground state of the Hubbard model

The circuit ansätze we consider are divided into layers; as the number of layers increases, the representational power of the ansatz increases. An initial test of the power of the variational method for producing ground states of the Hubbard model is to determine the number of layers required to produce a state $|\psi\rangle$ of 0.99 fidelity with the ground state $|\psi_G\rangle$, where

$$\text{Fidelity}(|\psi\rangle) = |\langle\psi_G|\psi\rangle|^2. \tag{3.7}$$

In Figure 3.7 we show this for the HV, EHV, and NPr ansätze. This illustrates that the EHV ansatz (which can be implemented efficiently) performs relatively well in comparison with the well-studied HV ansatz. In most cases (except the 2×3 grid), the HV ansatz requires a lower number of layers, but this is outweighed by the depth reduction per layer achieved by using the EHV ansatz. Note that in the case $N_x = 1$, the two ansätze are equivalent.

Figure 3.7 also demonstrates that the NPr ansatz generally requires lower depth than the other two ansätze to achieve high fidelity. This is expected, as it corresponds to optimising over a larger set of circuits. It also shows that the optimisation procedure does not experience significant difficulties with this larger set other than increased run time. This caused by the increased number of function evaluations required to optimise the larger set of parameters. This increase can be significant; for example, a 1×11 grid required approximately 10^5 function evaluations and a run time of 16.5 hours on a GPU-accelerated system to achieve fidelity 0.99 using the NPr ansatz, whereas achieving the same fidelity using the EHV ansatz required fewer than 9000 function evaluations and a run time of 1.5 hours.

Focusing on the EHV ansatz, in Figure 3.8 we illustrate how the fidelity improves with depth, for the largest grid sizes we considered. In each case, the infidelity (1 - fidelity) decreases exponentially with depth. Notably, 2×6 seems to be more challenging than 3×4.

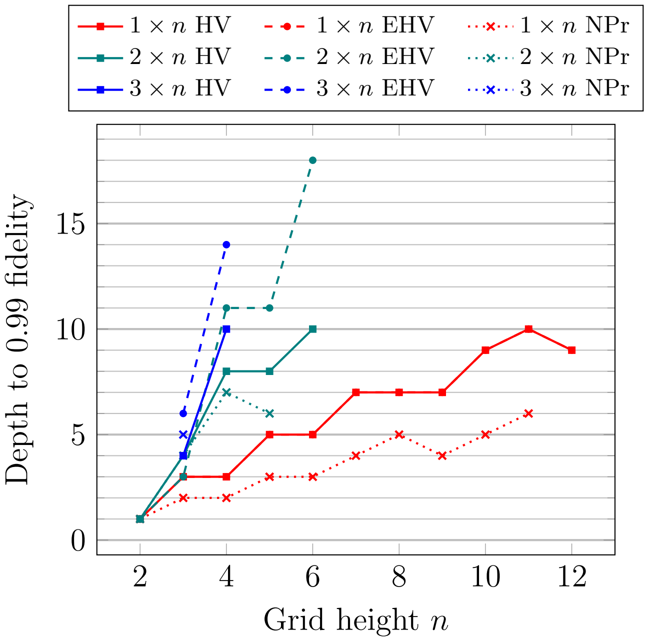


Figure 3.7: Ansatz depths required to represent the ground state of the $N_x \times N_y$ Hubbard model for the HV, EHV and NPr ansätze. Each point corresponds to the minimal-depth circuit instance we found (using the L-BFGS optimiser) that produces a final state with fidelity at least 0.99 with the true Hubbard model ground state ($t = 1$, $U = 2$). Tests were run for all grids of size $N_x N_y \leq 12$. For $1 \times n$ grids, HV and EHV are the same.

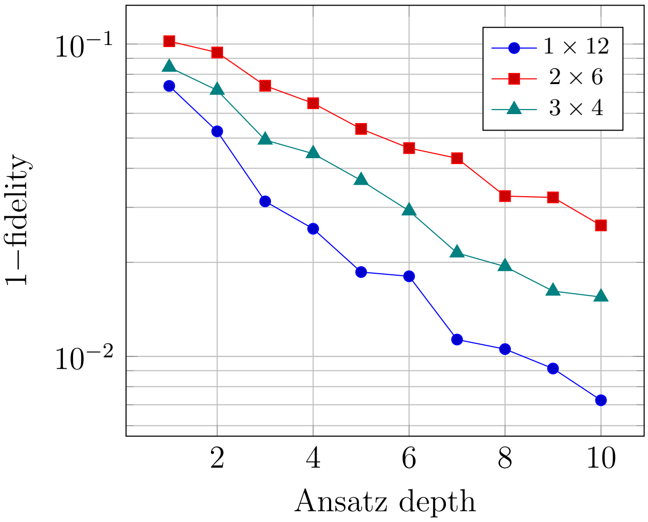


Figure 3.8: Scaling of infidelity (1−fidelity) with number of layers of the EHV ansatz for grids with 12 sites.

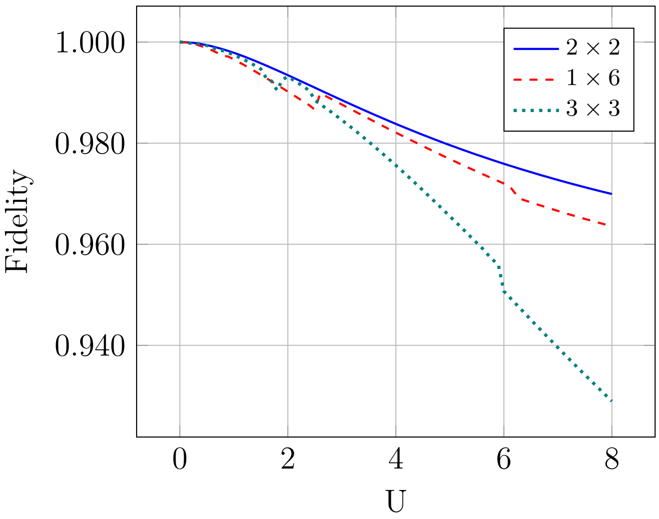


Figure 3.9: Final fidelity achieved with varying U for a 2×2 grid (at depth 1), 1×6 (at depth 5), and 3×3 (at depth 6), using the EHV ansatz, simulated exact energy estimates, and the L-BFGS optimiser. U incremented in steps of size 0.1.

Effect of choice of U parameter

For many of the simulations in this chapter, we fix the weight U of the onsite term in the H_{hub} to 2, as was also done in [92]. Here we will demonstrate how the performance of the EHV ansatz changes with U. We considered three grid sizes (2×2, 1×6 and 3×3), and evaluated the fidelity achieved for different choices of U at the same depth for which the $U = 2$ case achieves fidelity > 0.99. This gives a measure of the difficulty of finding the ground state. The results are shown in Figure 3.9, we can see that the fidelity decreases as U increases. However, the final fidelity achieved continues to be quite high for all $U \leq 4$.

Figure 3.10 demonstrates the minimal depth of the EHV ansatz required to reach 0.99 fidelity as U varies. In general the depth required increases with U, which is to be expected as we begin in the $U = 0$ ground state. As we can see from Figure 3.10, to get to the physically interesting intermediate coupling regime $U = 4$, where classical methods experience significant uncertainties [46, Table V], only requires one or two extra ansatz layers from $U = 2$. However, the more strongly correlated model $U = 8$ requires roughly double the ansatz layers.

We remark that, when optimising with realistic measurements, the statistical uncertainty in the energy estimates is likely to increase linearly with U. This is because the energy is estimated by summing several measurement results, some of which are scaled by a U factor. Thus going from $U = 2$ to $U = 8$ (for example) would

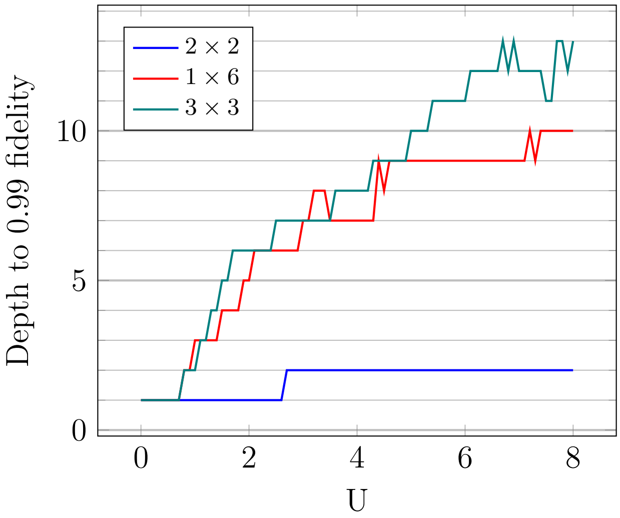


Figure 3.10: Depth of the EHV ansatz required to reach 0.99 fidelity with the ground state of the Hubbard model as a function of U with grid sizes 2×2, 1×6 and 3×3.

likely require 16 times more measurements to achieve the same level of statistical uncertainty.

The half-filled regime

While we are mostly concerned with finding the overall ground state of H_{hub}, solutions of certain restricted cases can be of interest as well. A prominent restriction is that of half-filling, where the number of fermions in the grid is exactly half of the number of sites. This case is easier to solve classically due to the lack of a sign problem [46], enabling quantum Monte Carlo methods to succeed. However, the special physical and mathematical characteristics of the half-filled regime make it an important benchmark for VQE methods

The performance of the EHV ansatz in terms of depth to high-fidelity solution can be seen in Figure 3.11; we can see that the depths required in the half-filling case are comparable to depths required to find the ground state of the full Hamiltonian (using the occupancies in Table 3.3).

In Figure 3.12, we see how the infidelity, absolute error with the actual ground state, and absolute error with the true double occupancy of the ground state changed with depth for an example grid size of 2×4, also at half-filling. While the optimisation is carried out by minimising the energy, we can see that the infidelity and the error in the double occupancy follow a very similar trend to the error in the ground energy. This gives us reason to believe that energy is a good figure-of-merit to op-

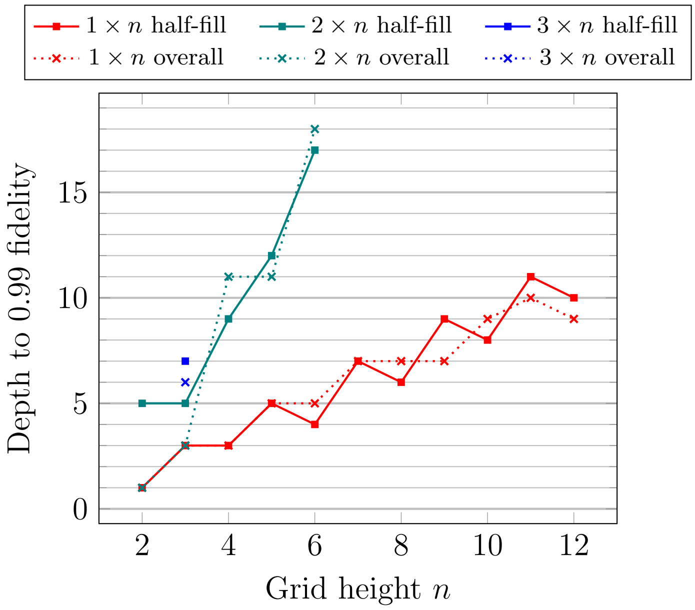


Figure 3.11: Depth of the EHV ansatz required to reach 0.99 fidelity with the ground state of the half-filled Hubbard model at $t = 1, U = 2$. Comparison with depth required to find the overall ground state (data reproduced from Figure 3.7).

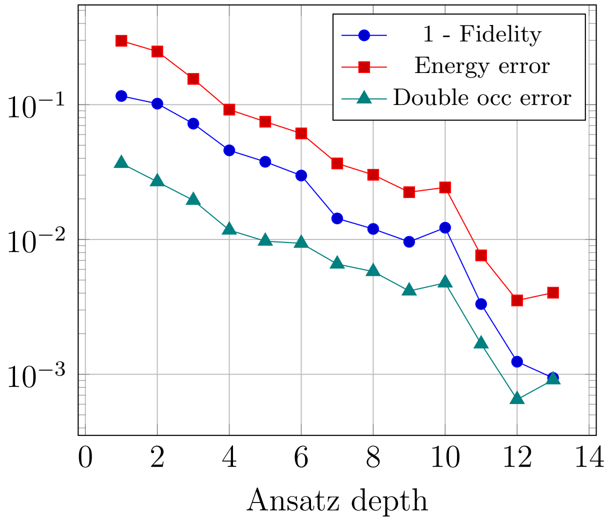


Figure 3.12: Infidelity, absolute error with the actual ground state and absolute error with the true double occupancy for various depths of the EHV ansatz for the 2×4 half-filled Hubbard model.

timise, even if a different property of the ground state (such as double occupancy) is of interest. The situation is similar away from half-filling.

Another peculiarity about the half-filled case is that degeneracy in the ground states of the non-interacting Hamiltonian, which is the initial state for the EHV ansatz, is common. If the degeneracy is low enough (only a few states), then trying out each of the degenerate states as the initial state might be feasible. However, in some of the grids with higher degeneracy we tried a few different solutions to arrive at a successful initial state. For the results presented in Figure 3.11, the initial states were generated as follows: if there was no degeneracy then the choice was the single ground state; for grid size 2×2, one of the hopping terms in the Hamiltonian was altered by $\epsilon = 0.0001$ allowing for a splitting between the degenerate states; for grid sizes 2×5, 3×3, a superposition over all the degenerate states was created and the weights of each of the states were added as parameters to be optimised over in the main optimisation; for all other degenerate states, a manual selection of initial state was carried out by trial-and-error.

3.2.2 Incorporating measurements

In this section we present results for the second level of realism, taking account of the statistical effect of measurements but assuming that we have an ideal quantum computer. We will compare the ability of SPSA (see Section 2.5.1) and CD (see Section 2.5.2) to find the ground state of the Hubbard model for four representative grid sizes. Before we present these results, we will first discuss optimiser choices.

As discussed in Section 2.5.1, we must specify values for the SPSA metaparameters; these can be tailored to the specific problem at hand. We fixed $\alpha = 0.602$ and $\gamma = 0.101$ to the recommended theoretical and practically effective values [118]. We chose the stability constant $A = 100$ to be around 10% of the maximum number of iterations. Finally, $a = 0.15$ and $c = 0.2$ were chosen by a joint parameter sweep (using SPSA to solve a small instance of the Hubbard model). We found that a and c generally had to be small to reduce the rate of convergence, which allowed us to reach a more accurate result but with more iterations.

In addition to selecting the values of the metaparameters, we made an extra modification to the SPSA algorithm where we perform multiple runs of the optimiser. We start with two coarse runs with a high level of statistical noise where we calculate the energy estimate using only 10^2 and then 10^3 energy measurements. This is followed by a finer run where SPSA is restarted using 10^4 energy measurements

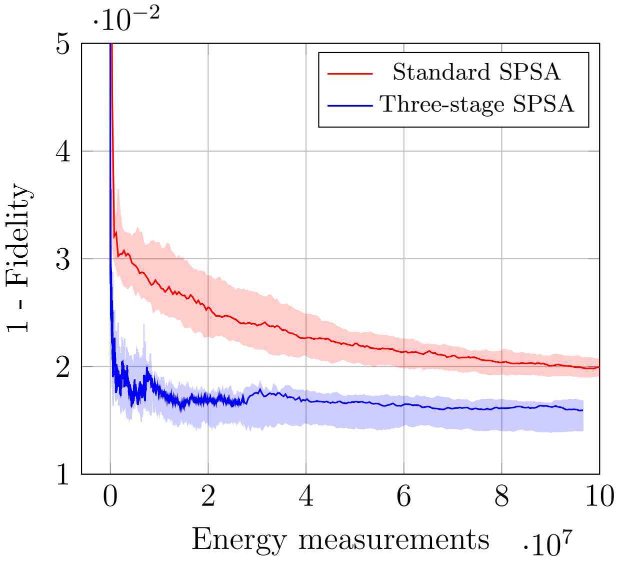


Figure 3.13: Infidelity achieved over five runs of the standard SPSA algorithm (where each energy estimate is formed of 10^4 energy measurements and two gradient evaluations are taken in each iteration) and a modified three-stage SPSA algorithm which starts with less accurate measurements, as described in the text. Results are shown for a 1×6 grid, EHV ansatz, depth 5. The solid lines show the median of the runs and the limits of the shaded regions are the maximum and minimum values seen over the five runs.

for the estimate and averaging over two gradient evaluations in random directions for $g(\boldsymbol{\theta}_k)$. The number of steps in this three-stage optimisation is determined by a ratio of $10:3:1$. Figure 3.13 shows the benefit of using coarse energy estimates near the start of the algorithm, so as to rapidly obtain a good approximation to the optimal value.

We now compare the ability of the three-stage SPSA and CD algorithms to find the ground state of Hubbard model instances for four representative grid sizes: 2×2, 1×6, 2×3, and 3×3. For CD, we fixed the number of approximate energy estimates to $\sim 1.2 \times 10^3$, where each estimate consists of 10^4 energy measurements. This translates to a limit of $\sim 6 \times 10^7$ circuit evaluations. For SPSA, the number of energy estimates was limited to $\sim 1.2 \times 10^4$, due to the number of measurements per estimate changing throughout the course of the optimisation. As described above, we carry out a three-stage optimisation routine and set the ratio of $10:3:1$ for very coarse, coarse, and smooth function evaluations, respectively. By limiting to a total of $\sim 1.2 \times 10^4$ energy estimates, we allow for a similar total limit as that of CD, $\sim 1.2 \times 10^7$ energy measurements (or $\sim 6 \times 10^7$ circuit evaluations).

For each grid size, we determined the final fidelity of the output of the VQE algo-

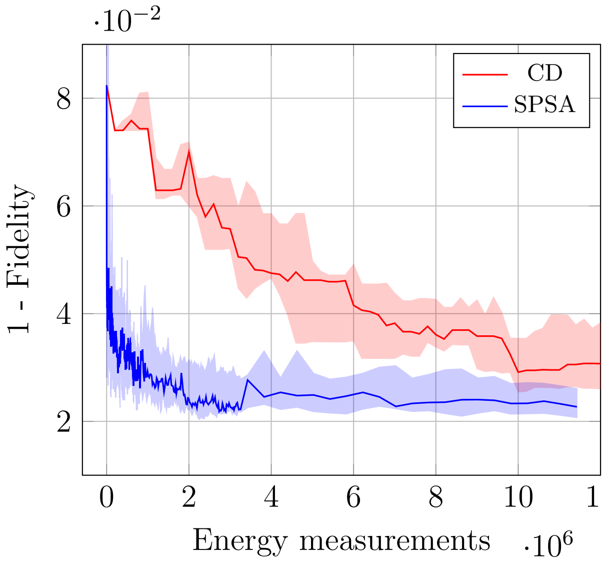


Figure 3.14: Infidelity reached during the optimisation process with CD and SPSA optimisers using realistic measurements. Results are shown for five runs of a 3×3 grid, using the depth 6 EHV ansatz. The solid lines show the median of the runs and the limits of the shaded regions are the maximum and minimum values seen over the five runs.

Grid	Depth	CD	SPSA	L-BFGS
2×2	1	0.0068	0.0066	0.0066
1×6	5	0.0293	0.0199	0.0098
2×3	3	0.0202	0.0199	0.0075
3×3	6	0.0307	0.0227	0.0068

Table 3.4: Final infidelity reached for CD and SPSA optimisers using realistic measurements, compared with the best infidelity achieved by the L-BFGS optimiser with exact estimates. The CD and SPSA results are the median of five runs.

rithm with the true ground state after the fixed number of measurements. We chose the minimal ansatz depth for which the ground state is achievable (via Figure 3.7). The results are shown in Figure 3.14 and Table 3.4. In all cases, both algorithms are able to achieve relatively high fidelity (considering that each energy measurement involves at most 10^4 circuit runs, suggesting an error of $\sim 10^{-2}$). However, in the case of 1×6 and 3×3 grids, SPSA achieves a noticeably higher fidelity. It is also interesting to note in Figure 3.14 that SPSA uses substantially fewer energy measurements to achieve a high fidelity. One reason for this may be that each iteration of CD requires more energy estimates ($2N_xN_y+1 = 19$ for a 3×3 grid, as compared with two energy estimates for SPSA).

		2 × 2		1 × 6		2 × 3	
p		CD	SPSA	CD	SPSA	CD	SPSA
10^{-3}	No ED	0.0066	0.0066	0.0262	0.0187	0.0231	0.0201
	ED	0.0066	0.0067	0.0297	0.0176	0.0174	0.0196
10^{-4}	No ED	0.0067	0.0066	0.0250	0.0188	0.0169	0.0194
	ED	0.0068	0.0066	0.0259	0.0180	0.0179	0.0195
10^{-6}	No ED	0.0065	0.0066	0.0288	0.0197	0.0174	0.0199
	ED	0.0064	0.0066	0.0257	0.0185	0.0183	0.0194

Table 3.5: Infidelities at end of runs for varying grid sizes and noise rates, with error detection (ED) off/on. The results presented are the median of three runs. The EHV ansatz at depths of 1, 5, 3 respectively was used.

3.2.3 Depolarising noise

Finally, we evaluate the effect of noise on the ability of the VQE algorithm to find the ground state of the Hubbard model. We considered a simple depolarising noise model where, after each two-qubit gate, each qubit experiences a Pauli error with probability p, which is either an X, Y or Z operation (chosen with equal probability) [12]. We examined noise rates $p \in \{10^{-3}, 10^{-4}, 10^{-6}\}$ for grid sizes 2×2, 1×6 and 2×3. These experiments are substantially more computationally costly than those with realistic measurements, because of the need the recompute the circuit (accounting for the depolarisation errors) for each energy measurement.

We also tested the effect of the simple error detection procedure described in Section 2.4. When an error is detected by the wrong Hamming weight being obtained, that run is ignored. The measurement procedure then continues until the intended number of valid energy measurements are produced for each type of term. Hence, the total number of energy measurements is somewhat larger than the noiseless case.

We list the final fidelities achieved for different grid sizes, error rates, and optimisation algorithms in Table 3.5. An illustrative set of runs for a 2×3 grid is shown in Figure 3.15. The overhead of error detection is not shown in this figure (that is, measurements where an error is detected are not counted). One can see that in all cases, errors due to depolarisation do not make a significant difference to the final fidelities achieved, compared with the noiseless results in Table 3.4. The use of error detection seems to usually lead to a small but noticeable improvement in the final fidelity achieved, as well as seeming to make the performance of the optimiser during a run less erratic. We note that error detection might have a more relevant

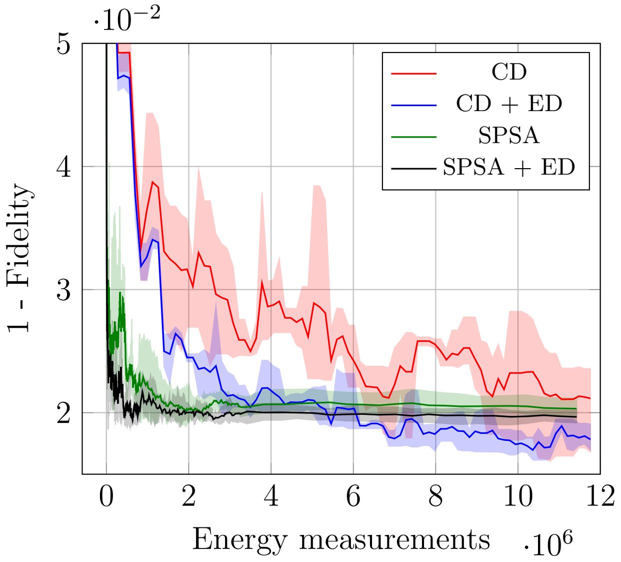


Figure 3.15: Infidelity reached during the optimisation process with CD and SPSA optimisers, with and without error detection. The results are for a 2×3 grid using the depth 3 EHV ansatz with an error rate of 10^{-3}. The solid lines show the median of the runs and the limits of the shaded regions are the maximum and minimum values seen over the 3 runs.

role for bigger grid sizes, due to higher depths and longer circuit run times.

Although our experiments indicate that low levels of depolarisation noise have little effect on the ansatz, it is necessary to look into more realistic noise models to fully investigate the effect of noise in real quantum devices.

3.3 Summary

We have carried out a detailed study of the complexity of variational quantum algorithms for finding the ground state of the Hubbard model. Our numerical results are consistent with the heuristic that the ground state of an instance on N sites could be approximately produced by a variational quantum circuit with $\sim N$ layers (and in all cases we considered, the number of layers required was at most $1.5N$).

If only around N layers are required, then the ground state of a 5×5 instance (larger than the largest instance solved classically via exact diagonalisation [50]) could be found using a quantum circuit on 50 qubits with around 25 layers. This corresponds to an approximate two-qubit gate depth of $24 + 25 \times (2 \times 5 + 2) + 1 = 325$ in a fully-connected architecture, including the depth required to produce the initial state. This is still beyond the capabilities of today's quantum computing hardware. Although we only considered relatively shallow quantum circuit depths, the ability of the NPr ansatz to find ground states suggests that the classical optimisation routines used could continue to work for these deeper circuits, as this ansatz used a much larger number of parameters, e.g. over 400 for the largest grids we considered.

In numerical experiments with simulated realistic energy measurements on systems with up to nine sites, the CD and SPSA algorithms were able to achieve high fidelity with the ground state (e.g. SPSA achieves fidelity > 0.977 for a 3×3 grid) by making a number of measurements which would require a few hours[3] of execution time on a real quantum computer. Including simulated depolarising noise in the quantum circuit for systems with up to six sites, error rates of up to 10^{-3} did not have a significant effect on the fidelity of the solution. These results are a step towards building confidence that the VQE algorithm could be effective on quantum hardware.

Determining the optimal choice of classical optimiser remains an important challenge. It is plausible that the optimisers used here could be combined or modified to improve their performance. Other methods that have been studied in this context include adaptive optimisation algorithms [113] and techniques based on machine learning [111, 112]. Future work should evaluate such methods for larger-scale instances of the Hubbard model and other challenging problems in many-body physics.

Variational methods show significant promise for producing the ground state of the Hubbard model for grid sizes somewhat beyond what is accessible with classical computational methods. Over the next two chapters, we will determine whether this is also the case when larger instances of the Hubbard model are first compressed using an embedding technique.

---

[3] $\sim$ 57M circuit evaluations; Google's Sycamore processor can perform 1M circuit evaluations in 200s [1].

Chapter 4

Density matrix embedding theory applied to the Hubbard model

Due to the limited number of qubits on NISQ devices, embedding algorithms which reduce the size of the problem Hamiltonian could be very useful. Methods such as density functional theory (DFT), dynamical mean-field theory (DMFT), density matrix renormalisation group (DMRG) and matrix product states (MPS), which have been used for decades in the classical simulation of solid-state systems, are gaining popularity in the quantum computing community [127–133].

The idea behind embedding methods is that the properties of a Hamiltonian H can be reproduced using a smaller embedded Hamiltonian. Density matrix embedding theory is one method for obtaining a suitable embedded Hamiltonian [134, 135] which has been used to study the Hubbard model [46, 134, 136, 137]. In the DMET algorithm a fragment of the original system is retained, with the rest of the system being mapped to a bath that is the same size as the fragment.

DMET is well suited for use with the VQE since it does not require the computation of any complicated time- or frequency-dependent quantities such as Green's functions. There have been a number of works over the past few years that have combined DMET with VQE. Rubin [138] investigated solving the 1D Hubbard model using a fragment containing one site with UCC as the VQE ansatz. Yamazaki et al. [139] conducted an analysis of DMET along with other embedded techniques for alkanes using classical quantum chemistry simulations to estimate qubit counts and sampling errors. More recently, experiments have been done on quantum hardware. Kawashima et al. [140] conducted an experiment on a trapped-ion quantum computer using an embedded Hamiltonian with two qubits to estimate the energy of a ring of hydrogen atoms. Tilly et al. [141] solved the Hubbard model on a

Bethe lattice using energy-weighted DMET with four qubits on IBM superconducting hardware.

Over the next two chapters we discuss in detail how the combination of DMET and VQE could be used to solve the Hubbard model on a quantum computer. We go beyond these small-scale experiments with a more systematic study of the topic, involving an analysis of the quantum circuit complexity and numerical simulations up to four sites (16 qubits).

In this chapter we focus on the classical aspects of the combined DMET and VQE algorithm. We start by discussing the idea behind DMET, followed by detailed steps of the single-shot embedding algorithm (a variant of DMET) applied to the Hubbard model. We finish the chapter with a derivation of the structure of the embedded Hamiltonian produced by single-shot embedding; this is key to determining the complexity of the VQE algorithm in the next chapter. This chapter is based on Section II and Appendices A and C from "Solving the Hubbard model using density matrix embedding theory and the variational quantum eigensolver" [59]. Original work is contained in Section 4.3, and additional details that were omitted from the paper are included there.

4.1 Introduction to DMET

In general, states of a quantum system can be written in terms of the basis states of two of its sub-systems. For our purposes, let us call the first sub-system F the fragment and the second sub-system E the environment. For example, for a system that consists of electrons in a grid, the fragment could be a subset of sites of the grid. Any state $|\Psi\rangle$ of the system can be written as

$$|\Psi\rangle = \sum_{i=1}^{N_F} \sum_{j=1}^{N_E} \Psi_{ij} |F_i\rangle |E_j\rangle, \tag{4.1}$$

where $|F_i\rangle, |E_j\rangle$ are basis states of F and E, and $N_{F/E}$ are the sizes of their respective Hilbert spaces. Using the singular value decomposition of Ψ_{ij} it can be rewritten as

$$\begin{aligned} |\Psi\rangle &= \sum_{i=1}^{N_F} \sum_{j=1}^{N_E} \sum_{\alpha=1}^{\min(N_F,N_E)} U_{i\alpha} \lambda_\alpha V^\dagger_{\alpha j} |F_i\rangle |E_j\rangle \\ &= \sum_{\alpha=1}^{N_F} \lambda_\alpha |F'_\alpha\rangle |B_\alpha\rangle, \end{aligned} \tag{4.2}$$

where without loss of generality we have taken $N_E > N_F$. The $|F_i\rangle$ states have been rotated to a new basis $|F'_\alpha\rangle = \sum_i U_{i\alpha}|F_i\rangle$ of the fragment. The $|B_\alpha\rangle = \sum_j V^\dagger_{\alpha j}|E_j\rangle$ are a reduced set of states, spanning a system called the bath. The bath represents the portion of the environment needed to model interactions with the fragment. This is the Schmidt decomposition of $|\Psi\rangle$ [142].

If $|\Psi\rangle$ were the ground state of a Hamiltonian H in the full system, then by construction it is also the ground state of a smaller embedded Hamiltonian H_{emb} given by

$$H_{\text{emb}} = \mathcal{P}^\dagger H \mathcal{P}, \tag{4.3}$$

with the projector $\mathcal{P}$ being

$$\mathcal{P} = \sum_{\alpha\beta} |F'_\alpha B_\beta\rangle\langle F'_\alpha B_\beta|. \tag{4.4}$$

In practice $|\Psi\rangle$ is not known so the exact embedding procedure cannot take place. Instead we look to approximate $\mathcal{P}$ by taking the Schmidt decomposition of another state $|\Phi\rangle$ which is determined self-consistently. Typically, $|\Phi\rangle$ is taken to be the ground state of a mean-field quadratic Hamiltonian H_{MF}, where H_{MF} is an approximation to H, as this can be calculated efficiently. Furthermore, depending on the variant of DMET, an alternative to equation (4.3) may be used to determine the embedded Hamiltonian from $\mathcal{P}$.

At the end of the self-consistency procedure, observables of H_{emb} (on the fragment and between the fragment and bath) are used to approximate observables of the full Hamiltonian H.

4.2 The single-shot embedding algorithm

There are many variants of the DMET procedure which choose different mean-field Hamiltonians H_{MF}, different ways of projecting onto the problem Hamiltonian H and different termination criteria for self-consistency. Here we have chosen to focus on the simplest form of DMET, single-shot embedding [142, 143]. This will highlight the key issues that would be associated with implementing any form of DMET on a quantum computer. Single-shot embedding has been shown to be effective in practice [139, 143] and has been successfully used with the VQE algorithm [138, 140] for one fragment site.

In Figure 4.1 we estimate the energy per site for the infinite 1D Hubbard model using single-shot embedding with fragment sizes N_{frag} of one and four (4 and 16

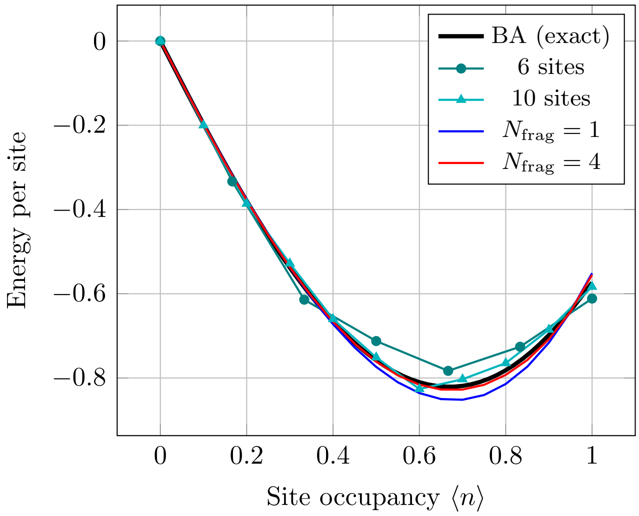


Figure 4.1: Energy per site for the 1D $U = 4$ Hubbard model solved with the Bethe ansatz (BA) [43, 44], exact diagonalisation for small models with periodic boundary conditions and single-shot embedding. Note that none of the calculations involve VQE.

qubits), and exact diagonalisation for small Hubbard models of 6 and 10 sites (12 and 20 qubits) with periodic boundary conditions. On a small quantum computer it may be preferable to run DMET as it can achieve a high accuracy with a small number of sites. It is also possible to input any fraction for $\langle n \rangle$ with DMET, whereas with small models the site occupancy will be restricted to values for which there are a whole number of electrons.

We will now lay out the steps in the single-shot embedding algorithm in the context of the Hubbard model. Recall from Section 1.3 that the Hubbard Hamiltonian is defined as

$$H_{\text{hub}} = -t \sum_{\langle i,j \rangle, \sigma} (a_{i\sigma}^\dagger a_{j\sigma} + a_{j\sigma}^\dagger a_{i\sigma}) + U \sum_i n_{i\uparrow} n_{i\downarrow} = T + W. \tag{4.5}$$

Here T describes the kinetic energy in the system, these are the hopping terms between nearest-neighbour sites. W describes the interactions between particles in the system, which are the onsite terms.

We will be considering the problem of finding properties of the ground state for the model on an infinite 1D or 2D rectangular grid that has a fixed fraction of the sites filled with electrons with the same proportion of spin-up and -down. In practice we will approximate the infinite grid by a large number of sites N with periodic or anti-periodic boundary conditions[1], and occupied by N_{occ} fermions split equally between spin-up and -down. The procedure to reduce this problem to an embedded Hamiltonian with N_{frag} sites in the fragment using single-shot embedding is as follows [142]:

1. *Calculate the ground state of the approximating mean-field Hamiltonian*
   The simplest form the mean-field Hamiltonian, H_{MF}, can take is the one-particle (quadratic) part of H_{hub},

   $$H_{MF} = T = -t \sum_{\langle i,j\rangle,\sigma} (a^\dagger_{i\sigma} a_{j\sigma} + a^\dagger_{j\sigma} a_{i\sigma}). \tag{4.6}$$

   Note that H_{MF} could be chosen to be different from T, which would lead to a different embedded Hamiltonian.

   We must find the one-particle reduced density matrix (1-RDM) of the ground state of H_{MF}. The 1-RDM expresses the relationship between the behaviour of an electron at two different sites and the matrix diagonal contains electron densities. For a state $|\psi\rangle$ it is defined to be

   $$\rho(|\psi\rangle)_{ij} = \langle\psi|a^\dagger_j a_i|\psi\rangle. \tag{4.7}$$

   H_{MF} is a quadratic Hamiltonian which can be solved efficiently and its ground state is a Slater determinant. This ground state can be found by taking the matrix C of coefficients of H_{MF}. Restricting to one spin-type since in this case both spins are identical, C is an $N \times N$ matrix with the $(i,j)^{\text{th}}$ element being the coefficient of $a^\dagger_{i\uparrow} a_{j\uparrow}$ in the Hamiltonian. Without loss of generality we assume that the orbitals have been ordered such that the environment sites follow the fragment sites.

   We then diagonalise C and put the eigenvectors corresponding to the lowest $N_{\text{occ}}/2$ eigenvalues (recall that half of the electrons are spin-up) into an $N \times N_{\text{occ}}/2$ matrix Φ which now represents the ground state Slater determinant (each column is an occupied orbital written as a linear combination of the original orbitals). Since Φ is a Slater determinant, its 1-RDM can be simply calculated as

   $$\rho(\Phi) = \Phi\Phi^\dagger. \tag{4.8}$$

   A derivation for this fact is provided in Appendix B.

---

[1] Periodic boundary conditions introduce $-ta^\dagger_{0\sigma} a_{N\sigma}$ terms into the Hamiltonian. Anti-periodic boundary conditions introduce $ta^\dagger_{0\sigma} a_{N\sigma}$.

2. *Construct the projector from the mean-field ground state*
   The 1-RDM of the ground state is used to construct the projector that will reduce the environment orbitals to the bath orbitals. If we delete the first N_{frag} rows and columns of the 1-RDM, we are left with an $N_{\text{env}} \times N_{\text{env}}$ submatrix (where $N_{\text{env}} = N - N_{\text{frag}}$) representing the environment orbitals. Diagonalising this submatrix leads to 3 different scenarios:

   - Eigenvalues of 0 correspond to unoccupied environment orbitals.
   - Eigenvalues of 1 correspond to occupied environment orbitals. Counting these tells us the occupation number of the embedded Hamiltonian we will need to solve in step 4. If there are m of these then the embedded occupation number including both spin-types is $N_{\text{emb}} = N_{\text{occ}} - 2m$.
   - Eigenvalues between 0 and 1 have overlap on the environment and the fragment. There will be N_{frag} of these and we will write the eigenvectors associated to them as v_i.

   The eigenvectors associated to eigenvalues of 0 and 1 are discarded and the rest are used to define the projector

   $$P = \begin{pmatrix} I & 0 \\ 0 & v_1 ... v_{N_{\text{frag}}} \end{pmatrix}, \tag{4.9}$$

   where I is the identity matrix of size $N_{\text{frag}} \times N_{\text{frag}}$. Note that this procedure outlined in steps 1 and 2 is equivalent to finding the Schmidt decomposition of $|\Phi\rangle$ to calculate the projector [142].

3. *Construct the embedded Hamiltonian from the projector*
   The embedded Hamiltonian H_{emb} is constructed using the non-interacting bath formulation [142] where only the quadratic part of H_{hub} is projected and higher order terms are only added back to the fragment. This is a simpler construction than projecting the full Hamiltonian H_{hub} which can lead to more complicated interaction terms.

   The projection of the quadratic part, T, of H_{hub} into the embedded basis is obtained as follows. Write $T = T^{\uparrow} + T^{\downarrow}$ (separating spin-up and -down terms) and interpret each term as a matrix of coefficients $K^{\uparrow}$ and $K^{\downarrow}$, similarly to C in step 1. Now project the matrices of coefficients into the embedded basis to obtain

   $$K^{\sigma}_{\text{emb}} = P^{\dagger} K^{\sigma} P, \quad \sigma \in \{\uparrow, \downarrow\}. \tag{4.10}$$

The K^{σ}_{emb} can then be re-interpreted as Hamiltonians T^{σ}_{emb}, which can be added up to obtain T_{emb}.

The two-particle interaction term in the embedded Hamiltonian is simply set to be the terms in W that act only on the fragment,

$$W_{\text{emb}} = U \sum_{i \in \text{frag}} n_{i\uparrow} n_{i\downarrow}. \tag{4.11}$$

Finally, a chemical potential μ that governs the number of electrons in the fragment is also added to the embedded Hamiltonian. This is the only parameter that is determined self-consistently in this variant of DMET (the "single-shot" refers to this single free parameter) and makes the embedded Hamiltonian

$$H_{\text{emb}} = T_{\text{emb}} + W_{\text{emb}} - \mu \sum_{i \in \text{frag},\sigma} n_{i\sigma}. \tag{4.12}$$

4. *Solve the embedded problem*

   H_{emb} is a Hamiltonian on $4N_{\text{frag}}$ orbitals ($2N_{\text{frag}}$ for each spin's fragment and bath sites). The ground state $|\Phi_{\text{emb}}\rangle$ of H_{emb} occupied by N_{emb} electrons can be found using methods such as exact diagonalisation, DMRG, or VQE.

5. *Adjust the chemical potential until there are the correct number of particles in the fragment*

   Repeat from equation (4.12) in step 3, adjusting μ until the fraction of occupied orbitals in the fragment matches the site occupancy of H_{hub}. Since μ is only one parameter, it can be fitted by finding roots of $f(\mu) = 0$ where

$$f(\mu) = \frac{N}{N_{\text{frag}}} \sum_{i \in \text{frag},\sigma} \langle \Phi_{\text{emb}} | n_{i\sigma} | \Phi_{\text{emb}} \rangle - N_{\text{occ}}. \tag{4.13}$$

   In the rest of this thesis we refer to this as the DMET function. $f(\mu)$ is equal to the number of electrons in the fragment scaled up to fill the large model, minus the number of electrons in the Hubbard model to be solved for.

More general forms of DMET have an extra optimisation loop. A correlation potential V is introduced in the mean-field Hamiltonian, giving $H_{MF} = T + V$, which is adjusted until the 1-RDMs of $|\Phi\rangle$ and $|\Phi_{\text{emb}}\rangle$ match [142]. In general DMET calculations, the system can also be split into multiple disjoint fragments, with the bath for each fragment being constructed from the union of the other fragments. Consistency then has to be enforced between all the separate fragment-bath systems [135, 142]. We do not need to consider this as the Hubbard model is translationally invariant. This makes the use of multiple fragments redundant as they would all have the same properties.

4.2.1 Calculating observables from the embedded Hamiltonian

Observables relevant to the original problem Hamiltonian H_{hub} can be calculated from the final $|\Phi_{\text{emb}}\rangle$ given by the single-shot embedding algorithm. The quantities of interest in this thesis are the energy and double occupancy per site. These are calculated by taking expectation values of $|\Phi_{\text{emb}}\rangle$ on the fragment and fragment-bath. Contributions purely from the bath are ignored.

For example, the energy of the fragment is calculated as [135, 136]

$$E_{\text{frag}} = \langle T^{\text{frag}}_{\text{emb}} \rangle + \frac{1}{2}\langle T^{\text{frag-bath}}_{\text{emb}} \rangle + \langle W_{\text{emb}} \rangle \tag{4.14}$$

where $T^{\text{frag}}_{\text{emb}}$ and $T^{\text{frag-bath}}_{\text{emb}}$ are the terms of T_{emb} that act on the fragment-only, or between the fragment and bath, respectively. The energy per site is then obtained by dividing by the number of sites in the fragment. Double occupancy of the fragment is calculated as [137]

$$D_{\text{frag}} = \sum_{i \in \text{frag}} \langle n_{i\uparrow} n_{i\downarrow} \rangle = \frac{\langle W_{\text{emb}} \rangle}{U}. \tag{4.15}$$

4.3 Form of the embedded Hamiltonian

This section contains a summary and derivation of the structure of the embedded Hamiltonian, which will be necessary for developing efficient swap networks and measurement schemes in Section 5.1. Unlike when using a classical procedure such as exact diagonalisation, having more terms in the embedded Hamiltonian results in a more complicated circuit being run on the quantum computer and requires more measurements to estimate the expectation values.

In general, the embedded Hamiltonian from equation (4.12) can be written more explicitly as

$$H_{\text{emb}} = \sum_{i \neq j \in \text{emb}, \sigma} t_{ij} (a^\dagger_{i\sigma} a_{j\sigma} + a^\dagger_{j\sigma} a_{i\sigma}) + \sum_{i \in \text{bath}, \sigma} t_{ii} n_{i\sigma} + U \sum_{i \in \text{frag}} n_{i\uparrow} n_{i\downarrow} - \mu \sum_{i \in \text{frag}, \sigma} n_{i\sigma}. \tag{4.16}$$

Determining the form of H_{emb} now comes down to knowing which terms are present in the Hamiltonian (non-zero t_{ij}). Here we will state the structure of the embedded

Hamiltonian when the single-shot embedding procedure is carried out for the 1D and 2D Hubbard models. These results are derived in Sections 4.3.1 and 4.3.2 by considering the matrix of coefficients of T and its projection T_{emb}. This derivation has been made possible due to the simple structure of T and properties of the Hubbard model such as translational invariance.

There are three different types of terms to consider – fragment-only terms, bath-only terms and fragment-bath hopping terms. The fragment-only terms retain the same structure as the Hubbard model and are nearest-neighbour hopping terms. For the fragment-bath interactions, each fragment site on the edge of the fragment shares a hopping term with all of the bath sites. In the 1D Hubbard model, the edge sites are the first and last sites of the fragment. For the 2D Hubbard model with a 1D fragment all of the fragment sites are on the edge.

For the terms acting only on the bath, the t_{ii} are generally all non-zero. With the 1D Hubbard model when using anti-periodic boundary conditions and taking the number of electrons of one spin-type ($N_{\mathrm{occ}}/2$) to be even (or with periodic boundary conditions and $N_{\mathrm{occ}}/2$ odd), the bath hopping terms in H_{emb} split into two groups – even- and odd-numbered sites. Within each of these two groups, every site has a hopping term with all the other sites. If these conditions are not met then all of the bath sites can interact with all of the other bath sites, increasing the number of interactions, as described in Section 4.3.1.

The embedded Hamiltonian of the 2D Hubbard model with a 1D fragment has the same structure of bath hopping terms as the 1D model. However when using a 2D fragment, the bath sites split into four groups where within each group all possible interactions occur. Unlike the 1D case there is no clear split (e.g. even/odd), but the size of the groups are roughly equal. Conditions on when this split occurs is discussed are Section 4.3.2.

From the standpoint of quantum circuit complexity, it will always be advantageous to use a 2D shaped fragment when solving the 2D Hubbard model. This is due to the fact that both the fragment-bath and bath-only hopping terms will be fewer in number than when using a 1D fragment shape.

4.3.1 1D Hubbard model

Here we will present a derivation for why the embedded Hamiltonian takes the form that it does when solving the 1D Hubbard model. In particular, we will explain why the bath hopping terms split into two groups, where in the first group all the even-numbered sites interact with each other, and in the second group odd-numbered sites interact. We will show that this occurs using periodic boundary conditions when $N_{\mathrm{occ}}/2$ is odd or using anti-periodic boundary conditions when $N_{\mathrm{occ}}/2$ even.

To do this we will be following through the first three steps of the single-shot embedding algorithm from Section 4.2. Throughout the derivation we will need to make use of $K^{\uparrow}$, the $N \times N$ coefficient matrix of the hopping terms T restricted to one spin-type. For simplicity of notation, we will refer to this matrix as T for the rest of this section. We will also take $t = 1$ in T and let $M = N_{\mathrm{occ}}/2$.

For T with periodic boundary conditions, we will show:

1. T is a circulant matrix. If M is odd then the 1-RDM will also be circulant (note that circulant matrices are also Toeplitz). The 1-RDM is symmetric by construction and so it is a symmetric Toeplitz matrix.

2. The properties of symmetric Toeplitz matrices imply that half of the eigenvectors of the 1-RDM placed in the projector P will be symmetric and half skew-symmetric.

3. When doing the matrix multiplication $T_{\mathrm{emb}} = P^{\dagger}TP$, the symmetric and skew-symmetric vectors cancel, leading to zeroes in the bath part of T_{emb}.

We will first present the details for the periodic case and then follow up with the anti-periodic case. Apart from the first step, the argument for the two cases is identical.

Periodic boundary conditions

Step 1

Let $\Phi = (v_0, v_1, \ldots, v_{M-1})$ where v_i are the eigenvectors of T associated to the lowest M eigenvalues. The 1-RDM can be written as

$$\rho = \Phi\Phi^{\dagger} = \sum_{i=0}^{M-1} v_i v_i^{\dagger} = \sum_{\lambda} P_{\lambda} \tag{4.17}$$

where the P_{λ} group together the $v_i v_i^{\dagger}$ where the v_i share the same eigenvalue λ. If the v_i contained in P_{λ} spans the full eigenspace of λ, then P_{λ} is a projector onto that eigenspace.

The matrix T is

$$T = \begin{pmatrix} 0 & -1 & & & -1 \\ -1 & \ddots & \ddots & 0 & \\ & \ddots & \ddots & \ddots & \\ & 0 & \ddots & \ddots & -1 \\ -1 & & & -1 & 0 \end{pmatrix}. \tag{4.18}$$

That is, T is the matrix with -1 on the off-diagonals and in the top-right and bottom-left corners, and zeros everywhere else. T is a circulant matrix which therefore commutes with the cyclic permutation matrix S where

$$S = \begin{pmatrix} 0 & & & 1 \\ 1 & \ddots & & \\ & \ddots & \ddots & \\ & & 1 & 0 \end{pmatrix}. \tag{4.19}$$

Commuting matrices preserve each other's eigenspaces and in particular commute with the projectors onto each other's eigenspaces. This means that S commutes with P_λ – and therefore ρ – provided that the P_λ project onto whole eigenspaces.

Whether the P_λ project onto whole eigenspaces depends on M, if there are eigenvalues with multiplicities greater than 1. To determine this we need to find the eigenvalues of T, which turns out to be a simple task since the eigenvalues/vectors of circulant matrices are well known [144]. The eigenvalues of T are given by

$$\lambda_j = -\omega^j - \omega^{(N-1)j} = -2\cos\left(\frac{2\pi j}{N}\right), \tag{4.20}$$

for $j = 0, \ldots, N-1$ and where $\omega = e^{2\pi i/N}$ is the Nth root of unity. There is one eigenvalue of -2 and one of 2 (if N is even), and the rest all come in pairs; hence M must be odd for the eigenvector pairs to be included in Φ.

From the discussion above, if M is odd, all the P_λ in ρ are projectors which commute with S, hence ρ commutes with S. A matrix is circulant if and only if it commutes with S [144], meaning that ρ is also circulant. Furthermore, a matrix of the form $\rho = \Phi\Phi^\dagger$ is always symmetric, and a circulant matrix is Toeplitz, therefore the 1-RDM is a symmetric Toeplitz matrix.

In fact, since T is circulant, we can go a step further and give an explicit formula for the 1-RDM. Eigenvectors of circulant matrices are given by

$$V_j = \frac{1}{\sqrt{N}} \begin{pmatrix} 1 & \omega^j & \omega^{2j} & \ldots & \omega^{(N-1)j} \end{pmatrix}^T \tag{4.21}$$

For $j = 0$ this is the (normalised) all-one vector. For $j > 0$, since $\lambda_j = \lambda_{N-j}$, the eigenspace associated to λ_j is spanned by $\{V_j, V_{N-j}\}$. Therefore it is possible to construct two eigenvectors from this that are real

$$\frac{V_j + V_{N-j}}{2} = \frac{1}{\sqrt{N}} \begin{pmatrix} 1 & \cos(\Omega j) & \cos(2\Omega j) & \dots & \cos((N-1)\Omega j) \end{pmatrix}^T , \tag{4.22}$$

$$\frac{V_j - V_{N-j}}{2i} = \frac{1}{\sqrt{N}} \begin{pmatrix} 0 & \sin(\Omega j) & \sin(2\Omega j) & \dots & \sin((N-1)\Omega j) \end{pmatrix}^T , \tag{4.23}$$

where $\Omega = 2\pi/N$. If we use these eigenvectors to construct Φ, then row a of Φ is equal to

$$\frac{1}{\sqrt{N}} \begin{pmatrix} 1 & \cos(a\Omega) & \sin(a\Omega) & \cos(2a\Omega) & \dots & \cos(\lfloor M/2 \rfloor a\Omega) & \sin(\lfloor M/2 \rfloor a\Omega) \end{pmatrix} . \tag{4.24}$$

The $(a, b)^{\text{th}}$ element of the 1-RDM ρ is obtained by multiplying row a by column b which gives

$$\begin{aligned} \rho_{ab} = (\Phi\Phi^\dagger)_{ab} &= \frac{1}{N} \left(1 + \sum_{k=1}^{\lfloor M/2 \rfloor} \cos(ka\Omega)\cos(kb\Omega) + \sin(ka\Omega)\sin(kb\Omega) \right) \\ &= \frac{1}{N} \left(1 + \sum_{k=1}^{\lfloor M/2 \rfloor} \cos(|a-b|k\Omega) \right) . \end{aligned} \tag{4.25}$$

Some intuition for why the elements of the 1-RDM only depend on the distance between sites a and b is that the Hubbard model is translationally invariant. Matrices with this property are symmetric Toeplitz.

Step 2

To calculate the projector P onto the embedded basis, we take the submatrix ρ_E of ρ that corresponds to the environment and calculate its eigenvalues/vectors. We then place the N_{frag} eigenvectors with eigenvalues between 0 and 1 into the projector according to equation (4.9). The rest of the eigenvectors have eigenvalues of either 0 or 1; these are discarded.

Since ρ_E is a symmetric Toeplitz matrix, $\lceil N_{\text{env}}/2 \rceil$ of its eigenvectors will be symmetric and $\lfloor N_{\text{env}}/2 \rfloor$ skew-symmetric[2]. This equal split of eigenvectors applies to the eigenspaces corresponding to multiple eigenvalues as well [145].

Therefore the eigenspaces associated to the eigenvalues of 0 and 1 will split as evenly as possible into symmetric and skew-symmetric, thereby leaving an equal split of eigenvectors for the eigenvalues between 0 and 1. This means that when the eigenvectors corresponding to the eigenvalues between 0 and 1 are placed in P, half of them will be symmetric and half skew-symmetric.

---

[2] Let J be the $N \times N$ matrix $J_{ij} = \delta_{i,N-1-j}$. A vector v is symmetric if $Jv = v$ and skew-symmetric if $Jv = -v$.

Step 3

We now project T using P to get T_{emb}.

$$\begin{aligned} T_{\text{emb}} &= P^{\dagger} T P \\ &= \begin{pmatrix} I & 0 \\ 0 & V^T \end{pmatrix} \begin{pmatrix} A_F & B \\ B^T & A_E \end{pmatrix} \begin{pmatrix} I & 0 \\ 0 & V \end{pmatrix} \\ &= \begin{pmatrix} A_F & BV \\ (BV)^T & V^T A_E V \end{pmatrix}, \end{aligned} \tag{4.26}$$

where the definitions of the submatrices are as follows. I is the identity matrix of size N_{frag} and 0 is the matrix of zeros. V is the matrix of eigenvectors of ρ_E that were placed in P, it is of size $N_{\text{env}} \times N_{\text{frag}}$ and since ρ_E is symmetric, V is real. $A_{F/E}$ are the matrices with -1s on the off-diagonal and are of size $N_{\text{frag}} \times N_{\text{frag}}$ and $N_{\text{env}} \times N_{\text{env}}$ respectively. B has a -1 in the bottom-left corner of the matrix and in the top-right.

It can be seen from equation (4.26) that A_F defines the fragment-only interactions, BV the fragment-bath interactions and $V^T A_E V$ the bath-only interactions. The hopping terms on the fragment have been preserved and are nearest-neighbour. Due to the structure of B, BV has zeros everywhere except the top and bottom row. This corresponds to the fragment sites on the ends of the fragment interacting with every bath site.

Finally, we turn to the bath-only interactions. Since A_E preserves the space of symmetric and skew-symmetric vectors, the columns of $A_E V$ are all symmetric or skew-symmetric. When the matrix multiplication $V^T A_E V$ is done, the symmetric rows of V^T will cancel with the skew-symmetric columns of $A_E V$ (and vice-versa), leading to zeroes in the bath part of T_{emb}. This corresponds to the bath sites splitting into two equal-sized groups where inside each group all of the sites share hopping terms. If we did not know that V contained symmetric and skew-symmetric eigenvectors then we could not show that the bath-only hopping terms have this structure, and $V^T A_E V$ could be completely dense.

We have observed in practice that when the eigenvectors in V are ordered according to their eigenvalues, they alternate symmetric and skew-symmetric. This is where the split into even and odd bath sites comes in. This property is called interleaving and in general is hard to prove, it is sufficient to just show the split into equal-sized groups for our purpose.

This analysis can be applied to any mean-field Hamiltonian that has a matrix of coefficients that is circulant (or almost-circulant; see the following section). One of the differences will be that the multiplicities of the eigenvalues of the Hamiltonian could lead to different restrictions on N_{occ}. Another will be the types of fragment-bath interactions that occur as this will depend on the form of B. For example, for the Hubbard model with next-nearest-neighbour interactions, the two fragment sites closest to each end will interact with all of the bath sites.

Anti-periodic boundary conditions

Step 1

Let T' be the matrix associated to the anti-periodic Hubbard model and S' a modified cyclic permutation matrix as follows,

$$T' = \begin{pmatrix} 0 & -1 & & & 1 \\ -1 & \ddots & \ddots & 0 & \\ & \ddots & \ddots & \ddots & \\ & 0 & \ddots & \ddots & -1 \\ 1 & & & -1 & 0 \end{pmatrix}, \quad S' = \begin{pmatrix} 0 & & & -1 \\ 1 & \ddots & & \\ & \ddots & \ddots & \\ & & 1 & 0 \end{pmatrix}. \tag{4.27}$$

T' is almost circulant but a minus sign is introduced when an element wraps back to the first column of the matrix. We will refer to this as an almost-circulant matrix. T' and S' commute.

Following a similar argument to before, S' will commute with ρ if the P_λ project onto whole eigenspaces. We will show that the eigenvalues of T' are given by

$$\lambda'_j = -2\cos\left(\frac{(2j+1)\pi}{N}\right) \quad \text{for } j = 0, \ldots, N-1 \tag{4.28}$$

which come in pairs (and if N is odd, an additional eigenvalue of 2), implying that M must be even for ρ to commute with S'.

Consider the principal submatrix T'_{N-1} of size $(N-1) \times (N-1)$ of T'. This matrix is tridiagonal and Toeplitz, therefore its eigenvalues[3] are distinct and given by $2\cos(\pi j/N)$ for $j = 1, \ldots, N-1$. Let v be the eigenvector associated to λ i.e. $T_{N-1}v = \lambda v$, and let v_{ext} be the vector obtained by lengthening v by one and setting the last component to zero. Let x be a column vector of length $N-1$ with 1 as its first element, -1 as its last, and zeros everywhere else. Then we have

---

[3] The eigenvalues of a tridiagonal Toeplitz matrix with a on the diagonal and b and c on the off-diagonals is well-known to be $a + 2\sqrt{bc}\cos(\frac{k\pi}{N+1})$ for $k = 1, \ldots, N$.

$$T'v_{\text{ext}} = \begin{bmatrix} T'_{N-1} & x \\ x^T & 0 \end{bmatrix} \begin{bmatrix} v \\ 0 \end{bmatrix} = \begin{bmatrix} T'_{N-1}v \\ v_0 - v_{N-2} \end{bmatrix} = \begin{bmatrix} \lambda v \\ 0 \end{bmatrix} = \lambda v_{\text{ext}} \tag{4.29}$$

if $v_0 = v_{N-2}$, i.e. if v is symmetric. According to Proposition 2 of [146], this means that eigenvalues of T'_{N-1} associated to symmetric eigenvectors are also eigenvalues of T' with multiplicity 2. The eigenvalues of T'_{N-1} associated to skew-symmetric eigenvectors are not eigenvalues of T'.

Furthermore, since T'_{N-1} is tridiagonal, symmetric and Toeplitz, it is possible to show that when its eigenvalues are arranged in descending order, they alternate symmetric and skew-symmetric. When $N-1$ is odd, the first eigenvector will be symmetric (since there are more symmetric than skew-symmetric vectors). When $N-1$ is even, since the off-diagonal is negative, the largest eigenvalue will be skew-symmetric [147]. Therefore, the symmetric eigenvectors of T'_{N-1} correspond to the eigenvalues given in equation (4.28), which are also the eigenvalues of T' with multiplicity 2.

Returning to the main part of the derivation, we must now determine what the structure of ρ is using the fact that it commutes with S', i.e. that $\rho = S'^T \rho S'$. We can match the right and left hand sides of this equation to get

$$\begin{pmatrix} \rho_{00} & \cdots & \rho_{0,N-2} & \rho_{0,N-1} \\ \vdots & \ddots & \vdots & \vdots \\ \rho_{N-2,0} & \cdots & \rho_{N-2,N-2} & \rho_{N-2,N-1} \\ \rho_{N-1,0} & \cdots & \rho_{N-1,N-2} & \rho_{N-1,N-1} \end{pmatrix} = \begin{pmatrix} \rho_{11} & \cdots & \rho_{1,N-1} & -\rho_{10} \\ \vdots & \ddots & \vdots & \vdots \\ \rho_{N-1,1} & \cdots & \rho_{N-1,N-1} & -\rho_{N-1,0} \\ -\rho_{01} & \cdots & -\rho_{0,N-1} & \rho_{00} \end{pmatrix}. \tag{4.30}$$

This implies that ρ is almost-circulant, and hence symmetric Toeplitz.

Step 2 and 3

The rest of the argument is now identical to the periodic case except that the matrix B has 1 in the top-right, not -1.

4.3.2 2D Hubbard model

Let T_n be the coefficient matrix of hopping terms for the n site 1D Hubbard model with (anti-)periodic boundary conditions. The hopping matrix T for the 2D $n \times m$ model is $T = T_n \otimes I_m + I_n \otimes T_m$, where I_n is the identity matrix of size n. If the eigenvalues of T_n are given by λ_{ni} and the eigenvectors by v_{ni}, then the eigenvalues of T are $\lambda_{ij} = \lambda_{ni} + \lambda_{mj}$ and the eigenvectors $v_{ij} = v_{ni} \otimes v_{mj}$.

Taking periodic boundary conditions as an example, it is clear to see that T commutes with $S_n \otimes S_m$ where S_n is the cyclic permutation matrix of size $n \times n$. Similar to the 1D case, we find that if whole eigenspaces are included in Φ then the 1-RDM also commutes with $S_n \otimes S_m$ and it turns out to be block circulant with circulant blocks (with the block sizes depending on n and m).

However, this structure does not remain in the submatrix ρ_E, making an analysis like the previous one very difficult. We observed that when the fragment shape is 1D, the bath sites split into odd and even groups. When the shape of the fragment is 2D, the bath sites split into four roughly equal groups and the grouping of terms changes with different input parameters to the model. These splits only happen when full eigenspaces are included in Φ. Since the multiplicities of the eigenvalues λ_{ij} depend on n and m, so do the allowed values of N_{occ}.

Chapter 5

The variational quantum eigensolver as an embedded system solver

In this chapter we turn our attention to the quantum aspects of the combined DMET and VQE algorithm. In the previous chapter we introduced the single-shot embedding algorithm, the simplest variant of DMET, which we will implement in this chapter. The VQE algorithm is used as a sub-routine to solve the embedded Hamiltonian H_{emb} within the single-shot embedding loop; in this chapter we investigate the details of the VQE algorithm.

While DMET reduces the number of qubits required to calculate ground state properties of a Hamiltonian, it leads to an embedded Hamiltonian with a more complicated structure than the original Hubbard Hamiltonian. Due to this, more complex ansatz circuits are required to find the ground state, and higher numbers of circuit preparations are needed to measure expectation values of the embedded Hamiltonian. To properly determine the quantum circuit complexity, we use the form of the embedded Hamiltonian derived in Section 4.3 to design optimised swap networks for the HV ansatz and construct efficient measurement schemes in Section 5.1. We follow this theoretical work with extensive numerical simulations in Section 5.2 with two levels of realism: assuming we can directly extract the exact expectation values, and simulating realistic measurements to represent an ideal quantum computer.

This chapter is based on Sections III-V and Appendices D and E from "Solving the Hubbard model using density matrix embedding theory and the variational quantum eigensolver" [59], with an extra discussion about the choice of SPSA meta-

Hubbard	Fragment	Ansatz depth	Measurements
1D	1D	$N_{\text{frag}}+3$	$N_{\text{frag}}+2$
2D	1D	$2N_{\text{frag}}$	$2N_{\text{frag}}$
2D	2D	$N_{\text{frag}}+N_E+N_x-2$ $+(N_y-4)\left\lceil\frac{N_x-4}{2}\right\rceil$	$N_{\text{frag}}+N_E$

Table 5.1: Number of layers of two-qubit gates required to implement one layer of the ansatz, and circuit preparations needed to measure all of the terms in the embedded Hamiltonian. N_{frag} is the DMET fragment size and corresponds to an embedded system with $4N_{\text{frag}}$ qubits. For a 2D fragment we take $N_{\text{frag}} = N_x \times N_y$ where we assume $N_x \leq N_y$. $N_E = 2(N_x + N_y - 2)$ is the number of sites on the edge of the 2D fragment.

parameters included in Section 5.2.2.

5.1 VQE implementation details

We will once again make use of the HV ansatz (see Section 2.3.1), which was shown to be effective for solving the Hubbard model in Chapter 3. The initial state for this ansatz will be the ground state of the non-interacting ($U = 0$) part of the embedded Hamiltonian H_{emb}, which can be efficiently prepared using Givens rotations [100]. The ansatz itself will consist of parametrised evolutions according to the hopping, onsite and number terms in H_{emb}.

As was done in Section 3.1.1 for the Hubbard model, in Section 5.1.1 we will perform an analysis of the two-qubit gate depth required to implement the HV ansatz circuit using swap networks on a fully-connected quantum computer. In Section 5.1.2 we present procedures for reducing the number of circuit preparations needed to obtain a measurement of $\langle H_{\text{emb}}\rangle$; this is analogous to Section 3.1.2 for the Hubbard model. The results of these analyses are summarised in Table 5.1 for combinations of the Hubbard model dimension and rectangular shaped fragments.

5.1.1 Efficient ansatz circuit implementation

In the HV ansatz, we alternate evolutions according to the onsite, hopping and number terms in H_{emb}. Similarly to Section 3.1.1, the onsite terms can all be implemented in one layer of two-qubit gates as they act on disjoint pairs of qubits.

The number operator terms are one-qubit gates. To determine the complexity of implementing the hopping gates, we will make use of fermionic swap networks [66].

We can restrict our analysis of implementing the hopping terms to one spin-type since the two spins are identical, therefore we consider swap networks on $2N_{\text{frag}}$ qubits (N_{frag} fragment and N_{frag} bath for one spin-type). From the generic swap network described in [66] and Section 2.3.2, we have an upper bound of $2N_{\text{frag}}$ layers of two-qubit gates since this accounts for all possible hopping interactions. If we use the structure of the embedded Hamiltonian it is possible to reduce the number of layers required.

Swap network for 1D model

Here we present a scheme for the 1D model where all the hopping interactions are done in $N_{\text{frag}}+2$ layers, giving almost a factor of 2 improvement compared with the upper bound for N_{frag} large.

Recall from Section 4.3 that when H_{hub} is one-dimensional, H_{emb} has the following hopping terms:

- **Fragment-only:** Nearest-neighbour, like the Hubbard model.
- **Fragment-bath:** The fragment sites on the edge (i.e. the first and last fragment site) interact with all of the bath sites.
- **Bath-only:** The bath sites split into two groups of odd- and even-numbered sites, where all the sites within a group interact with each other.

Note that the lower bound on the layers of two-qubit gates required to implement all of these hopping gates is $N_{\text{frag}}+1$, since each fragment site on the edge needs to interact with its one neighbouring fragment site and all N_{frag} bath sites.

Let F_i denote fragment site i and B_i bath site i. We choose a JW ordering for one spin-type such that the fragment-edge sites F_0 and $F_{N_{\text{frag}}-1}$ start close to the bath, and the even/odd bath sites are placed next to each other. Let the ordering be

$$F_{N_{\text{frag}}-3}\,F_{N_{\text{frag}}-4}\cdots F_2\,F_1\,F_0\,F_{N_{\text{frag}}-2}\,F_{N_{\text{frag}}-1}\;B_1\,B_3\cdots B_{N_o}\;B_0\,B_2\cdots B_{N_e}, \tag{5.1}$$

where if N_{frag} is even, $N_o = N_{\text{frag}}-1$ and $N_e = N_{\text{frag}}-2$ and vice-versa if N_{frag} is odd.

At the first layer of the swap network the hopping term between $F_{N_{\text{frag}}-1}$ and $F_{N_{\text{frag}}-2}$ is carried out. For the following N_{frag} layers, combined FSWAP and hopping

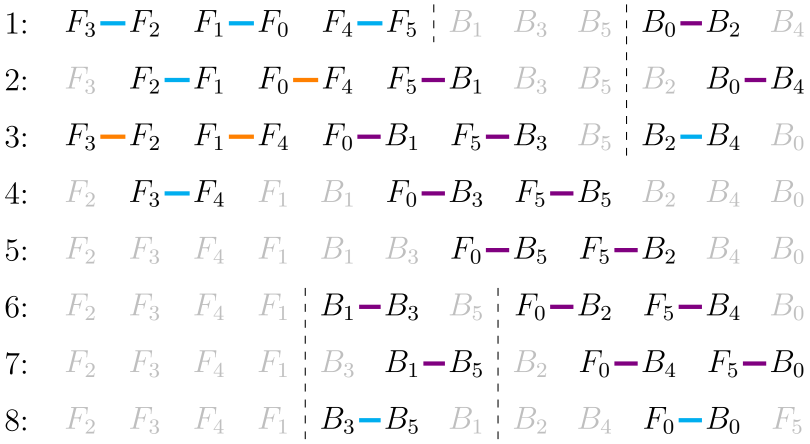

1: F3 F2 F1 F0 F4 F5 B1 B3 B5 B0 B2 B4
2: F3 F2 F1 F0 F4 F5 B1 B3 B5 B2 B0 B4
3: F3 F2 F1 F4 F0 B1 F5 B3 B5 B2 B4 B0
4: F2 F3 F4 F1 B1 F0 B3 F5 B5 B2 B4 B0
5: F2 F3 F4 F1 B1 B3 F0 B5 F5 B2 B4 B0
6: F2 F3 F4 F1 B1 B3 B5 F0 B2 F5 B4 B0
7: F2 F3 F4 F1 B3 B1 B5 B2 F0 B4 F5 B0
8: F2 F3 F4 F1 B3 B5 B1 B2 B4 F0 B0 F5

Figure 5.1: Demonstration of the swap network for the 1D model using $N_{\text{frag}} = 6$. The blue lines are hopping gates, the orange are fermionic swap gates and the purple are combined FSWAP and hopping gates between two neighbouring qubits. The dashed lines are added to aid the eye and unused qubits are greyed out.

gates are done between $F_{N_{\text{frag}}-1}$ and the bath sites to its right. This implements all of the hopping terms for the fragment-edge site $F_{N_{\text{frag}}-1}$. Simultaneously, a similar procedure can be carried out for the other edge site F_0. At the first layer the hopping term between the pair (F_0, F_1) is implemented, at the second layer an FSWAP is done between F_0 and $F_{N_{\text{frag}}-2}$ to place F_0 next to the bath sites, and for the following N_{frag} layers F_0 interacts with all the bath sites through combined FSWAP and hopping gates. Therefore, $N_{\text{frag}} + 2$ layers of two-qubit gates are required to implement all of the hopping gates associated to the fragment-edge sites.

The remaining fragment- and bath-only hopping terms can be fitted within these $N_{\text{frag}}+2$ layers. All of the fragment hopping terms can be implemented in two layers (see Section 3.1.1) except for $(F_{N_{\text{frag}}-2}, F_{N_{\text{frag}}-3})$, which need to be placed adjacent to each other first. After the second layer of the swap network, there are $N_{\text{frag}} - 4$ qubits between them, requiring $\lceil (N_{\text{frag}} - 4)/2 \rceil$ layers of FSWAPs to bring the two sites next to each other in the JW ordering.

The hopping terms between the even bath sites can be implemented in $\lceil N_{\text{frag}}/2 \rceil$ layers using the generic swap network from [66]. $F_{N_{\text{frag}}-1}$ takes $\lfloor N_{\text{frag}}/2 \rfloor + 1$ layers to interact with its neighbouring fragment site and the odd bath sites, so the entire even bath swap network can fit in these layers. Similarly, the odd bath swap network requires $\lfloor N_{\text{frag}}/2 \rfloor$ layers of two-qubit gates and these can all fit in the layers after they have interacted with F_0.

Figure 5.1 demonstrates this full procedure with fragment size $N_{\text{frag}} = 6$. Note that this swap network leaves the qubits in a less structured order. To get the same

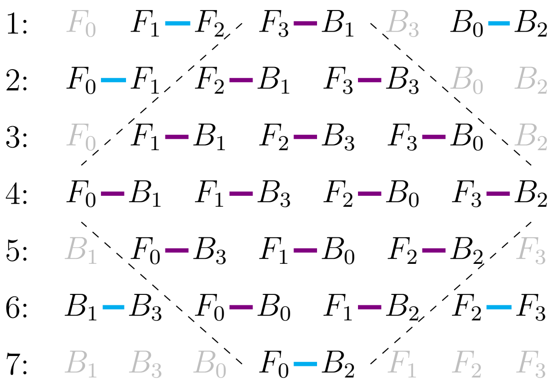

1: F_0 F_1—F_2 F_3—B_1 B_3 B_0—B_2
2: F_0—F_1 F_2—B_1 F_3—B_3 B_0 B_2
3: F_0 F_1—B_1 F_2—B_3 F_3—B_0 B_2
4: F_0—B_1 F_1—B_3 F_2—B_0 F_3—B_2
5: B_1 F_0—B_3 F_1—B_0 F_2—B_2 F_3
6: B_1—B_3 F_0—B_0 F_1—B_2 F_2—F_3
7: B_1 B_3 B_0 F_0—B_2 F_1 F_2 F_3

Figure 5.2: Demonstration of the swap network for the 2D model using a 1D fragment of size four. The blue lines are hopping gates and the purple are combined FSWAP and hopping gates between two neighbouring qubits. The dashed lines enclose all the fragment-bath interactions.

two-qubit gate depth of $N_{\text{frag}} + 2$, we simply reverse the swap network at the next ansatz layer.

Finally, incorporating the onsite and number gates brings the total two-qubit gate depth for one complete layer of the ansatz to $N_{\text{frag}} + 3$. All the onsite gates need an additional layer to complete. The number operator terms are one-qubit gates which can act on idle qubits in the swap network.

Swap network for 2D model

When H_{hub} is 2D but the shape of the fragment is 1D, recall from Section 4.3 that the structure of H_{emb} is similar to that of the 1D model except now all of the fragment sites interact with all of the bath sites. Consider two sets A and B of N_A and N_B qubits. If every qubit in set A has a hopping term with every qubit in B and the JW ordering has all the qubits in A followed by B, then $N_A + N_B - 1$ layers of FSWAP gates are required to do all the interactions by swapping the qubits in A through B. As a result, $2N_{\text{frag}} - 1$ layers of two-qubit gates are required to implement all the necessary hopping terms (mapping A to the fragment and B to the bath).

If the JW ordering has all fragment sites followed by the odd bath sites followed by all the even bath sites, all the fragment-bath interactions for one spin-type can be done in $2N_{\text{frag}} - 1$ layers. The fragment- and bath-only hopping terms can then fit within these layers. This is demonstrated in Figure 5.2 for a fragment size $N_{\text{frag}} = 4$.

The more complicated (and interesting) case is when the shape of the fragment is also 2D. We start with a reminder of the hopping terms:

- **Fragment-only:** Nearest-neighbour horizontal and vertical, like the Hubbard model.

- **Fragment-bath:** The fragment sites on the edge interact with all of the bath sites.

- **Bath-only:** The bath sites split into four groups of approximate size $N_{\text{frag}}/4$, where all the sites within a group interact with each other.

Let us take the fragment size to be $N_{\text{frag}} = N_x \times N_y$ and assume that $N_x \leq N_y$. The fragment sites will be ordered with the snake ordering [45, 100] (see Section 3.1) and will be followed by the bath sites placed in their four groups. We could use a different ordering, but this has the advantage of simplicity and enables us to make use of the efficient swap network for the Hubbard model described in Section 3.1.1.

The general structure of the swap network is as follows. The bath sites will be swapped through the fragment sites (where the bath sites come across a fragment-edge site a combined FSWAP and hopping gate will be done, otherwise just an FSWAP), and simultaneously fragment-edge sites will be moved along the snake towards the incoming bath sites. During this, the nearest-neighbour hopping terms will be implemented using the efficient swap network designed for the Hubbard model. The four sets of bath site hopping terms will be implemented using the generic swap network [66].

To determine the complexity of the circuit, we must consider the different components that make up the network separately.

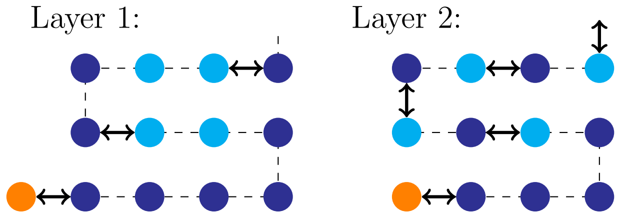
- **Fragment-only:** The fragment hopping terms (nearest-neighbour horizontal and vertical) can be completed in $2N_x$ or $2N_x + 1$ layers of two-qubit gates for N_x even or odd respectively; see Section 3.1.1.

- **Fragment-bath:** If all the N_E fragment-edge sites where

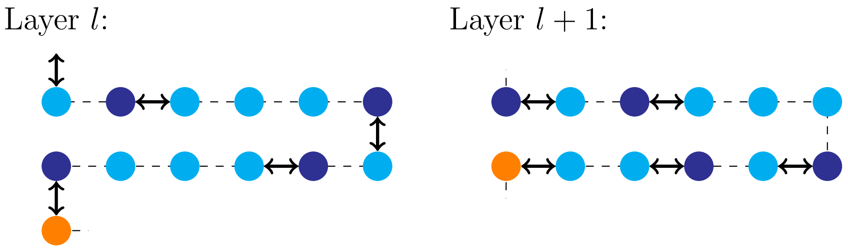
  $$N_E = 2(N_x + N_y - 2) \tag{5.2}$$

  are placed next to the N_{frag} bath sites, then $N_{\text{frag}} + N_E - 1$ layers of two-qubit gates are required to carry out all the fragment-bath interactions (using the procedure described for the 2D model with a 1D fragment).

- **Bath-only:** If the four groups of bath sites are of size $N_i \approx N_{\text{frag}}/4$ for $i = 1, 2, 3, 4$ then each of their individual swap networks can complete in N_i layers [66].



(a) First two layers of the swap network for $N_x = 4$.



(b) Selected layers of the swap network for $N_x = 6$.

Figure 5.3: Sections of the swap network for a 2D fragment. The dark blue circles represent fragment-edge sites, the light blue circles represent middle fragment sites and the orange circles are bath sites. The dotted line shows the JW snake ordering; quantum gates should only happen along this line. When acting between a bath site and a fragment-edge site, the arrows represent combined FSWAP and hopping gates; otherwise they represent FSWAP gates.

We now combine these three different swap networks together in as few layers as possible. Since the fragment-bath interactions will require the most layers, let us first consider how many will be required when the fragment starts in the snake ordering. For $N_x \leq 4$, it is possible to use FSWAPs to move the middle fragment sites away in time so that they never come into contact with the bath sites, as can be seen in Figure 5.3a. In this case all the fragment-bath interactions can be done in the minimum $N_{\text{frag}} + N_E - 1$ layers of two-qubit gates.

For $N_x > 4$, the pattern that the fragment sites settle into is: edge, middle, edge, $N_x - 3$ middle sites, repeated. When a bath site comes across this set of $N_x - 3$ middle sites, it is not possible to swap them away in time – see Figure 5.3b. Every time this occurs an extra $\lceil (N_x - 4)/2 \rceil$ layers of FSWAPs is done to bring the bath site to the next fragment-edge site. This happens $N_y - 4$ times during the entire swap network since the row of all fragment-edge sites at the beginning and end of the fragment creates a buffer of edge sites. This leads to $(N_y - 4) \lceil (N_x - 4)/2 \rceil$ extra layers of two-qubit gates, as compared to the $N_x \leq 4$ case.

The next thing to check is whether the bath- and fragment-only interactions can fit within these layers, or if more will be required. Since $N_i \approx N_{\text{frag}}/4$, three of the bath swap networks can complete before they interact with the fragment sites, and the final one after its associated bath sites have passed through the fragment.

This leaves the fragment hopping terms which require $2N_x$ layers to complete (taking N_x to be even for this argument). In this time the bath sites can interact with all the sites in the first two rows of the fragment, which means that all the horizontal and vertical hopping terms for the third row of the fragment onwards can be implemented before they come into contact with bath sites. However, the vertical hopping terms between the first and second rows of the fragment will need to be carried out after they have swapped through the bath sites, requiring an extra $N_x - 2$ middle fragment sites to pass through all of the bath sites. This adds $N_x - 2$ layers to the swap network[1], making the final count of layers

$$L = N_{\text{frag}} + N_E + N_x - 3 \tag{5.3}$$

for $N_x \leq 4$, and

$$L + (N_y - 4)\left\lceil \frac{N_x - 4}{2} \right\rceil \tag{5.4}$$

for $N_x > 4$.

At the next ansatz depth the swap network described here is reversed, allowing us to get the same circuit complexity at every depth.

5.1.2 Measurement scheme

As discussed in Section 2.4, all of the onsite and number terms can be measured in one circuit preparation by measuring every qubit in the computational basis. All the hopping terms on n qubits can be measured in n circuit preparations by diagonalising using the operator M (see equation (2.23)) and measuring non-crossing hopping terms simultaneously. The number of circuit preparations required to measure all of the terms of H_{emb} is given in Table 5.1.

Measuring the 1D model

By taking into account the structure of H_{emb} we can reduce the number of circuit preparations required. Restricting to one spin-type, it is possible to measure the hopping terms of H_{emb} for the 1D Hubbard model in $N_{\text{frag}} + 1$ circuit preparations

---

[1] Despite the fragment swap network for N_x odd requiring an extra layer, this does not translate into an extra layer for the 2D embedded swap network. This is because the JW snake ordering can be chosen such that the vertical hopping term that requires this extra layer is between the first two rows.

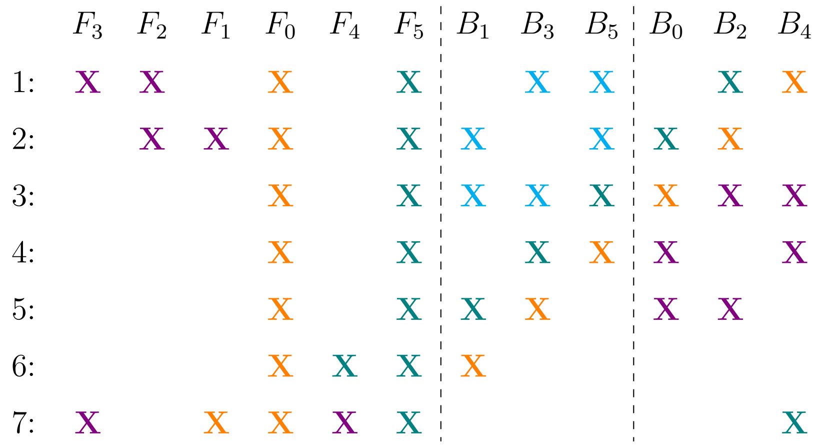


Figure 5.4: Demonstration of the measurement pattern for the 1D model using a fragment of size six. The numbered rows are different preparations of the circuit. On each row, Xs of the same colour represent a hopping term that has been measured, with the different colours showing which terms can be measured simultaneously. Note that M is applied on Xs of the same colour and then computational basis measurements are done on all the qubits.

(rather than $2N_{\text{frag}}$ using the rainbow scheme from Section 2.4). This is the minimum bound possible since the two fragment-edge sites each interact with $N_{\text{frag}} + 1$ other sites.

Assuming the ordering in equation (5.1), in the first circuit preparation we measure the terms (F_0, B_{N_e}) and $(F_{N_{\text{frag}}-1}, B_{N_e-2})$. At subsequent preparations, we measure the fragment-bath terms by working our way down the bath sites, i.e. at the second preparation we measure (F_0, B_{N_e-2}) and $(F_{N_{\text{frag}}-1}, B_{N_e-4})$. It is clear to see that in this way all of the hopping terms on the fragment-edge sites can be measured in $N_{\text{frag}} + 1$ runs of the circuit. The measurement of the remaining fragment- and bath-only terms can fit within these runs – considering the measurement of the odd bath hopping terms using the general rainbow procedure described in Section 2.4, this takes $\lfloor N_{\text{frag}}/2 \rfloor$ preparations and can be completed before the term $(F_{N_{\text{frag}}-1}, B_{N_o})$ is measured. This is best demonstrated in Figure 5.4 with fragment size six as an example.

Measuring the 2D model

Before we discuss how to measure all of the hopping terms for the 2D model, we turn to the situation where there are two sets A and B of $N_A \leq N_B$ qubits where every qubit in set A shares a hopping term with every qubit in set B. Assuming the JW ordering places A before B, then all the hopping terms can be measured in

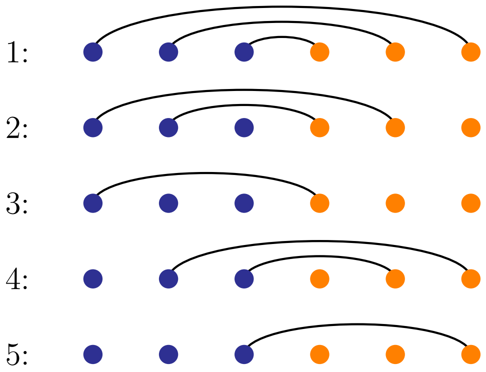


Figure 5.5: Measurement of all hopping terms between qubits in set A (blue) and qubits in set B (orange) in $N_A + N_B - 1$ circuit preparations.

$N_A + N_B - 1$ circuit preparations, saving one layer compared to the rainbow scheme. The measurement of the fragment-bath hopping terms is covered by this scenario.

At the first circuit preparation we measure all of the qubits in A with the furthest N_A qubits in B. At subsequent preparations we make our way down through the qubits in B until the first qubit in A has been measured with the first qubit in B. This requires N_B circuit preparations. For the remaining $N_A - 1$ preparations, we switch and measure the last qubits in B with the furthest possible qubits in A (that have not already been measured). This is shown in Figure 5.5 for $N_A = N_B = 3$.

Turning back to H_{emb}, we assume that the fragment- and bath-only hopping terms can be measured within the runs required for the fragment-bath terms. Using the procedure described above, the fragment-bath terms can be measured in $N_{\text{frag}} + N_E - 1$ preparations where N_E is the number of sites on the edge of the fragment.

It is possible to reduce the number of circuit preparations if we allow for swapping the qubit ordering around. Let us focus on the 2D model with a 1D fragment. Using the procedure described above, we require $2N_{\text{frag}} - 1$ circuit preparations to measure all of the hopping terms. By changing the JW ordering of the bath sites, it is possible to measure all of the terms in $\lceil 3N_{\text{frag}}/2 \rceil$ preparations.

We initially keep the JW ordering fixed (all fragment sites, followed by odd then even bath sites). All of the fragment- and bath-only hopping terms can be measured in $\lceil N_{\text{frag}}/2 \rceil$ preparations. The fragment-only terms can be measured in two preparations and the odd and even bath-only terms can be measured in $\lfloor N_{\text{frag}}/2 \rfloor$ and $\lceil N_{\text{frag}}/2 \rceil$ preparations respectively; these can all be done in parallel. At the next circuit preparation, we measure N_{frag} fragment-bath hopping terms with one rainbow between the first fragment site and the last bath site (in the JW ordering). At the next preparation, we change the JW ordering of the bath sites so that they all occupy a different position, and then do the same large rainbow between the first and last qubits. This continues for another $N_{\text{frag}} - 2$ preparations. In this way, all of the fragment-bath interactions are measured in N_{frag} preparations, leading to a total of $\lceil 3N_{\text{frag}}/2 \rceil$ for all the hopping terms in H_{emb}.

This change in JW ordering can be achieved by two different methods. We could directly change the JW ordering after the ansatz circuit by using FSWAP gates. Depending on what the order is changed to, this takes between two and N_{frag} layers of two-qubit gates. This extra requirement is not ideal for NISQ devices; depending on the device, it may be preferable to do more runs of the quantum computer instead. The other option is to change the JW ordering before the circuit is implemented on a quantum computer [94]. There are no extra requirements on the circuit depth and we also require less runs of the quantum computer. However, numerical validation would be required to check that the ansatz performs in the same way; much like the difference between the HV and EHV ansatz observed in Section 3.2.1.

5.2 Numerical results

In this section we present results from the numerical simulations that were done combining DMET with VQE. We ran numerical simulations up to a fragment size of four (16 qubits) for a range of parameters of the 1D and 2D Hubbard models using exact values for the expectation $\langle \psi(\boldsymbol{\theta}) | H_{\text{emb}} | \psi(\boldsymbol{\theta}) \rangle$, this is discussed in Section 5.2.1. We ran simulations up to a fragment size of two using measurements to estimate the expectation, this takes statistical noise into account; this will be covered in Section 5.2.2. These are the first two levels of realism considered in Section 3.2.

We will describe the details of the optimiser used for the VQE algorithm in their respective sections, but will mention the method used to find the roots of the DMET function in equation (4.13) here. We use the secant method which can be thought of as an approximation to Newton's method using finite-differences. The recurrence relation used in the secant method is

$$x_n = \frac{x_{n-2} f(x_{n-1}) - x_{n-1} f(x_{n-2})}{f(x_{n-1}) - f(x_{n-2})}, \tag{5.5}$$

where $f(x)$ is the DMET function. Two initial values x_0 and x_1 are needed, and termination criteria such as the maximum number of iterations or the accuracy required (e.g. distance $f(x_n)$ is from 0) need to be specified. We found that the

Term type	HV-min	HV-max
Onsite	1	N_{frag}
Number	1	$2N_{\text{frag}}$
Hopping	$N_{\text{frag}} + N_E - 1$	Fragment-only: $N_{\text{frag}} - 1$ Fragment-bath: $N_E N_{\text{frag}}$ Bath-only: $I(\lceil N_{\text{frag}}/2 \rceil) + I(\lfloor N_{\text{frag}}/2 \rfloor)$
Total	$N_{\text{frag}} + N_E + 1$	$4N_{\text{frag}} + N_E N_{\text{frag}} + I(\lceil N_{\text{frag}}/2 \rceil) + I(\lfloor N_{\text{frag}}/2 \rfloor) - 1$

Table 5.2: Parameter counts for one layer of the HV-min and -max ansätze for 1D fragment shapes. When solving the 1D Hubbard model, $N_E = 2$ and for 2D, $N_E = N_{\text{frag}}$. $I(n) = n(n-1)/2$ is the maximum number of hopping interactions between n qubits. Note that the number of parameters for HV-min hopping is the same as the number of circuit preparations required to measure all of the hopping terms – see Section 5.1.2.

secant method often converged within a few iterations and was an effective root-finding technique for our purpose.

The code was written in C++ and QuEST [56] was used to simulate the quantum circuits. Simulations for 16 qubits were run on the Google Cloud platform using an Nvidia Tesla P4 GPU. Smaller sizes were run on a laptop with an Intel i7-8th gen CPU.

5.2.1 Exact simulations

We ran two variants of the HV ansatz – one with a low number of parameters per layer (equal to the minimal number of sets of commuting terms in H_{emb}), and one with a high number of parameters per layer (equal to the number of terms in H_{emb}, but affixing the same parameters to identical spin-up and -down terms). We call these ansätze HV-min and HV-max. The number of parameters per ansatz layer for HV-min is $O(N_{\text{frag}})$ and for HV-max is $O(N_{\text{frag}}^2)$. Table 5.1 gives a breakdown of the parameter counts for a 1D shaped fragment.

Simulations were run up to an ansatz depth of 10 and 5 for HV-min and -max respectively. At each layer of the ansatz, we ran the onsite gates followed by the hopping gates and then number gates. Gates were implemented in the order that they would be in the swap network, including the reversal of the circuit at every other layer to fairly represent the behaviour of the actual quantum circuit that would be implemented on hardware.

		N_{frag} (Min params)				N_{frag} (Max params)			
U	$\langle n \rangle$	1 (3)	2 (5)	3 (6)	4 (7)	1 (4)	2 (11)	3 (18)	4 (25)
1	0.5	1	1	1	1	1	2	1	1
	1	1	2	1	1	1	2	1	1
2	0.5	1	2	4	2	1	2	2	2
	1	1	3	5	6	1	2	2	3
4	0.5	1	4	5	5	1	2	3	3
	1	2	4	8	>10	2	3	4	>5
8	0.5	2	5	6	8	2	2	3	4
	1	2	7	>10	>10	2	3	5	>5

Table 5.3: Depth of the ansatz required to achieve 1% relative error against the ground energy per site, calculated with exact diagonalisation as the DMET solver for the 1D model using the HV ansatz with minimum and maximum number of parameters. The number of parameters is shown in brackets.

		N_{frag} (Min params)				N_{frag} (Max params)			
U	$\langle n \rangle$	1 (3)	2 (5)	3 (7)	2×2 (8)	1 (4)	2 (11)	3 (20)	2×2 (32)
1	0.5	1	1	1	1	1	1	1	1
	1	1	2	1	2	1	1	1	1
2	0.5	1	2	1	1	1	2	2	2
	1	1	2	1	3	1	2	1	2
4	0.5	1	2	1	2	1	2	2	2
	1	2	3	4	5	2	2	2	4
8	0.5	2	3	1	4	2	2	2	3
	1	2	5	7	9	2	3	5	>5

Table 5.4: Depth of the ansatz required to achieve 1% relative error with the ground energy for the 2D model.

The L-BFGS optimisation algorithm provided by the nonlinear optimisation library NLopt [126] was used for the VQE classical optimiser as it was found to be effective in Chapter 3. For finding the root of the DMET function, we set the secant method to terminate when $|f(x)| < 0.1$.

The infinite 1D and 2D Hubbard models were approximated with a 240 site and 20×24 site finite size model with anti-periodic boundary conditions. Simulations were run with fragment sizes of 1, 2, 3 and 4. In the 2D case with fragment size

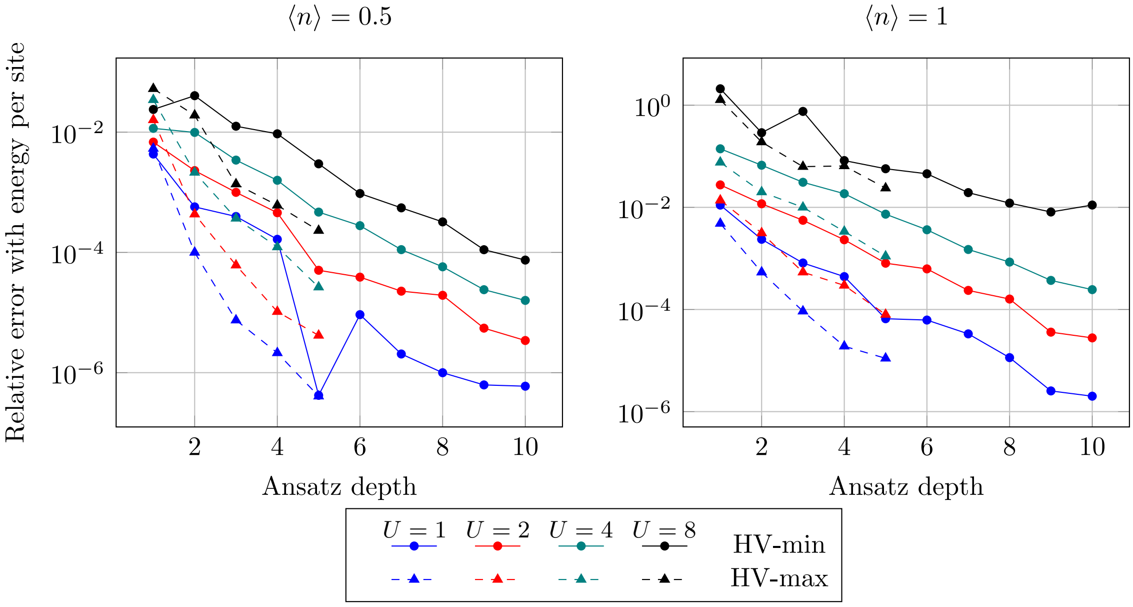


Figure 5.6: Comparison of the HV-min and -max ansätze for the 2D model with the largest fragment size of 2 × 2 for quarter- and half-filling. The relative error is against the ground energy per site calculated with exact diagonalisation as the DMET solver.

$N_{\text{frag}} = 4$ we took the fragment shape to be 2×2 and did not consider 1×4. Recall that a fragment size of N_{frag} requires $4N_{\text{frag}}$ qubits, so we simulated systems containing up to 16 qubits.

For each fragment size we fixed $t = 1$ and initially ran experiments for $U = 1$, 2, 4 and 8 with quarter- and half-filling, which correspond to a fermion site occupancy of $\langle n \rangle = 0.5$ and 1. For each experiment using VQE as the DMET solver, we ran a corresponding one using exact diagonalisation to compare the two solvers.

Tables 5.3 and 5.4 show the ansatz depths required to reach a 1% relative error with the energy per site when using exact diagonalisation as the solver. The calculation of the energy per site at the end of the DMET algorithm is stated in Section 4.2.1. The HV-max ansatz requires less depth to reach the same error than HV-min. However, this comes at a cost with the classical optimiser needing extra circuit evaluations. For $N_{\text{frag}} = 2$, the optimiser took 5-10× more evaluations for the same ansatz depth. For the larger fragment sizes this went up to 10-20× with some extreme cases requiring up to 100× more circuit runs.

For $N_{\text{frag}} = 1$, due to its small size, the behaviour of the ansatz as the depth increases was different from the larger fragment sizes. At depth one, the error was typically on the order of 10^{-1}-10^{-3} (depending on the value of U and $\langle n \rangle$). This error dropped to 10^{-6}-10^{-8} for depth two and plateaued for the other depths, meaning there is no benefit to going beyond depth two for the HV ansatz with $N_{\text{frag}} = 1$. This is not the case for the other fragment sizes as increasing the depth almost always resulted in a lower error – an example of this can be seen in Figure 5.6 for a fragment size of 2×2.

We found that the depth required increased as U increased, which is to be expected as the starting state for the HV ansatz is the ground state for the $U = 0$ embedded Hamiltonian. This can also be seen in Figure 5.6 for solving the 2D model with a 2×2 fragment. The features of these two graphs – that depth required increases as U increases, that the depth required is higher for half-filling than quarter-filling and that HV-max requires roughly 2-3 fewer layers to get to the same accuracy as HV-min – are representative of all the fragment sizes larger than one. The VQE algorithm is a nested optimisation loop providing imperfect solutions to H_{emb} within the larger DMET optimisation loop. The fact that the error in the energy per site goes down exponentially with the number of ansatz layers is an encouraging sign that the combination of DMET and VQE is effective.

Producing observables

After running the batch of experiments discussed above, we carried out further experiments to compute physical properties of the Hubbard model. Figure 5.7 is a plot of energy per site against site occupancy for the 1D model for $U = 1, 4, 8$ and N_{frag} =1, 2, 4. The VQE ansatz used was HV-min and the depths chosen for the graph were those required for each U at half-filling to reach 1% error (see Table 5.3). The 1D Hubbard model is exactly solvable using the Bethe ansatz [43, 44] and has been plotted as a reference. The lines reproduce the behaviour seen using DMET in the original paper from Knizia and Chan [134].

A plot of double occupancy per site against U is shown in Figure 5.8 for the 1D model for quarter- and half-filling and N_{frag} =1, 2, 4. In the case of half-filling, the double occupancy curve is not reproduced well even when using a fragment size of four. However, this deficiency is also present when using exact diagonalisation as the single-shot embedding solver, and is not a consequence of using VQE.

For completeness, we also plot the energy and double occupancy curves for the 2D model in Figure 5.9. These cannot be compared to exact values as these are not available for the 2D Hubbard model, but may be compared to data from other numerical methods [46]. The lines for the different fragment sizes are more bunched

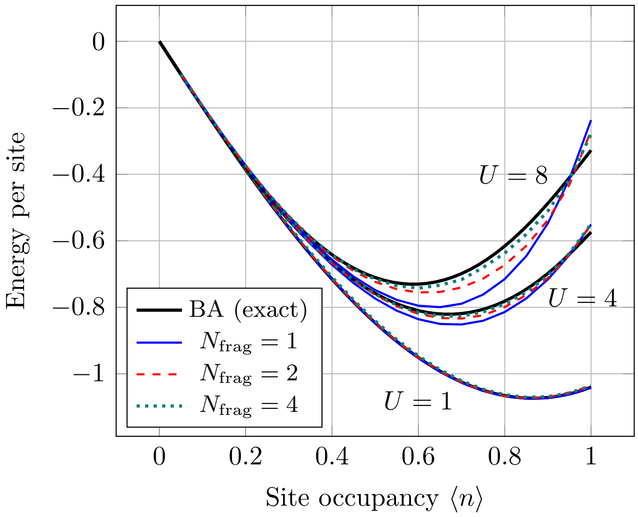


Figure 5.7: Plot of energy per site against site occupancy for the 1D model. The black line is the exact solution to the 1D Hubbard model calculated using the Bethe ansatz. The coloured lines are the values found using VQE as the DMET solver with 20 points taken between $\langle n \rangle = 0.05$ and 1. The ansatz used was HV-min. For $N_{\text{frag}} = 1$, the VQE depth used was 2, and for the other fragment sizes for each line we used the depths required for the given U at half-filling in Table 5.3 (or depth 10 in the cases where 1% relative error was not reached).

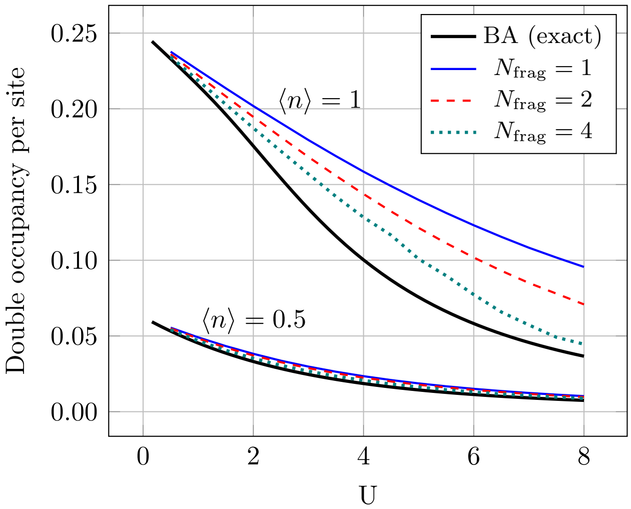


Figure 5.8: Plot of double occupancy per site against U for the 1D model. The coloured lines are the values found using VQE as the DMET solver with 16 points taken between $U = 0.5$ and 8. The ansatz used was HV-min. For $N_{\text{frag}} = 1$, the VQE depth used was 2, and for the other fragment sizes we used the depths required for 1% relative error with the double occupancy at $U = 8$, which was generally the same as the depths in Table 5.3 except for $N_{\text{frag}} = 4$, $\langle n \rangle = 0.5$ where it was depth 10.

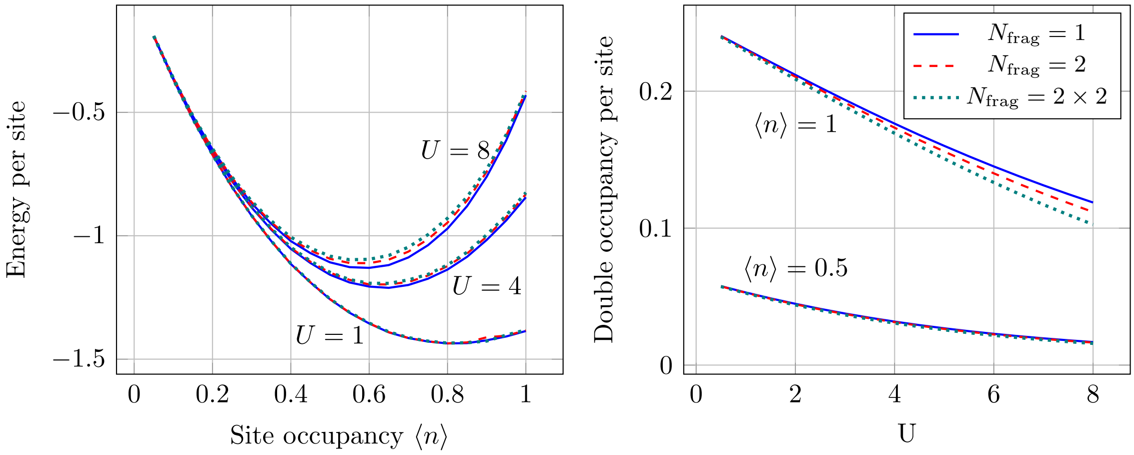


Figure 5.9: Plot of energy per site against site occupancy and double occupancy per site against U for the 2D Hubbard model. The ansatz used was HV-min. For $N_{\text{frag}} = 1$, the VQE depth used was 2. For the other fragment sizes for each line we used the depths required for the given U at half-filling in Table 5.4, while plotting the energy curve. For the double occupancy curve, we used the depths required for 1% relative error with the double occupancy at $U = 8$, which was generally the same as the depths in Table 5.4, with the exceptions being: $N_{\text{frag}} = 2, \langle n \rangle = 0.5$ (depth 4) and $N_{\text{frag}} = 2, \langle n \rangle = 0.5$ (depth 6).

than for the 1D Hubbard model, showing less of a difference for running DMET with small fragment sizes.

5.2.2 Incorporating measurements

We have shown that the VQE algorithm performs well as the solver for DMET when exact values are taken for the expectation values. Using exact values is a good test bed for trying out different ansätze and checking if the algorithm can work in principle, but we also need to consider the more practical aspects of quantum computers such as measurements and noise. Here we run simulations which include sampling from the quantum computer. We do not consider any type of noise, hence our simulations represent an ideal quantum computer.

The repeated measurement of states was simulated by storing the probability amplitudes of the state vector and then sampling from that discrete distribution; see Section 6.3. We picked a few representative simulations from the previous section to re-run and used the SPSA optimisation algorithm [116, 118] in place of L-BFGS. SPSA is a form of stochastic gradient descent where a gradient is taken in one random direction (instead of all directions); it is designed to be robust to

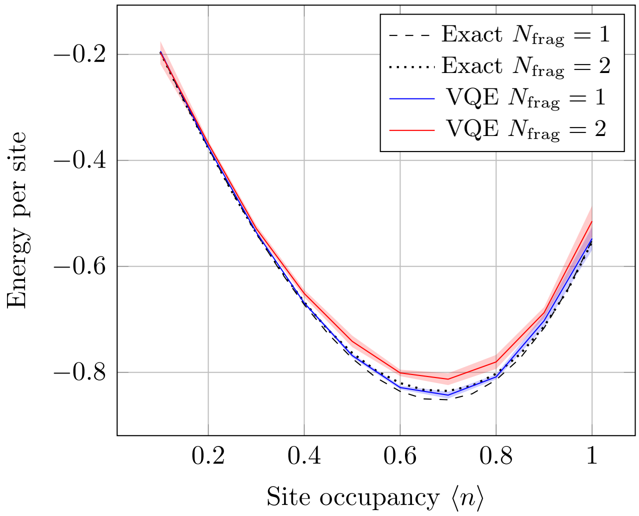


Figure 5.10: Plot of energy per site for the $U = 4$ 1D Hubbard model solved with $N_{\text{frag}} = 1$ and 2 using exact diagonalisation and VQE (with measurements) as the solver. The ansatz circuit is HV-min and the VQE depth is 2 for $N_{\text{frag}} = 1$ and 4 for $N_{\text{frag}} = 2$. Ten points were taken between $\langle n \rangle = 0.1$ and 1. The solid lines show the mean of ten DMET runs, with the shaded region being the standard deviation.

noise and require fewer function evaluations (see Section 2.5.1 for more details). In Section 3.2.2 we showed that SPSA was effective when solving the Hubbard model using VQE.

We picked the $U = 4$ 1D Hubbard model and ran the HV-min ansatz up to a fragment size of two with a range of fillings. The SPSA meta-parameters [118] were set to be $\alpha = 0.602, \gamma = 0.101$ (the theoretically optimal values [116]); $c = 0.2$ (from Section 3.2.2); and $a = 2, A = 10$ (to allow for fast convergence, and justified later on in this section). Each term in the expectation $\langle H_{\text{emb}} \rangle$ was estimated using 10^4 samples and the final state at the end of the SPSA algorithm with 10^5 samples. The maximum number of SPSA iterations was set to be 2,000 for $N_{\text{frag}} = 1$ and $10,000$ for $N_{\text{frag}} = 2$. As before we use the secant method to find μ in the DMET optimisation loop but loosen the termination criteria to stop if $|f(x)| < 0.5$.

Figure 5.10 is a plot of the energy per site against site occupancy when using VQE with sampling as the DMET solver. Solving H_{emb} with fragment size one using VQE reproduces the exact diagonalisation results with on average 0.5-1.5% relative error. However the fragment size two VQE curve has a larger relative error of around 2-5%. The fidelities of the ground states output from SPSA with the (exact) ground state of H_{emb} were typically above 0.999 for $N_{\text{frag}} = 1$ and around 0.985-0.995 for $N_{\text{frag}} = 2$. It is likely that by changing the SPSA meta-parameters or by using a different optimisation method, the fidelity for fragment size two could be increased.

In addition to the problem of whether the optimisation method chosen for VQE will converge, we must also consider whether the root-finding technique used to find μ will be able to handle the extra statistical noise. Despite the secant method having to deal with a noisier function, we found that in practice it still performed well most of the time and usually converged in less than 10 iterations. As a test for this we bypassed the secant method by solving H_{emb} with the optimal value of μ using SPSA and found that the plot of the energy per site was similar to Figure 5.10.

Occasionally we found that the secant method became unstable and did not converge. This happened quite rarely (for example for one filling and one run out of the 10 runs) and so we re-ran the simulations where this occurred. When running on quantum hardware this could be dealt with by monitoring which values of μ the secant method picks or by averaging multiple runs of SPSA for a certain μ. Other possibilities include combining the secant method with a curve fitting technique so that all of the known information about the DMET function can be effectively used, or squaring the function and using a gradient descent algorithm to find the minimum.

Choice of SPSA metaparameters

We finish this section with an example that demonstrates why it is important to tailor the SPSA metaparameters for different problems. In particular, we will discuss why the parameters governing step size were changed to $a = 2, A = 10$, compared to $a = 0.15, A = 100$ when solving the Hubbard model in Section 3.2.2.

Consider the embedded Hamiltonian for fragment size one. This acts on 4 qubits which we can label as follows: qubit 0 – spin-up fragment, qubit 1 – spin-up bath, qubit 2 – spin-down fragment, qubit 3 – spin-down bath. The embedded Hamiltonian has the form

$$H_{\text{emb}} = H_{\uparrow} + H_{\downarrow} + U n_0 n_2, \tag{5.6}$$

where

$$H_{\uparrow} = -\mu n_0 + \alpha(a_0^{\dagger} a_1 + a_1^{\dagger} a_0) + \beta n_1, \tag{5.7}$$

with $H_{\downarrow}$ identical to $H_{\uparrow}$ but on qubits 2 and 3. $\alpha, \beta \in \mathbb{R}$ are determined by the single-shot embedding algorithm described in Section 4.2 and μ is the chemical potential that is found using a root-finding technique in the DMET loop.

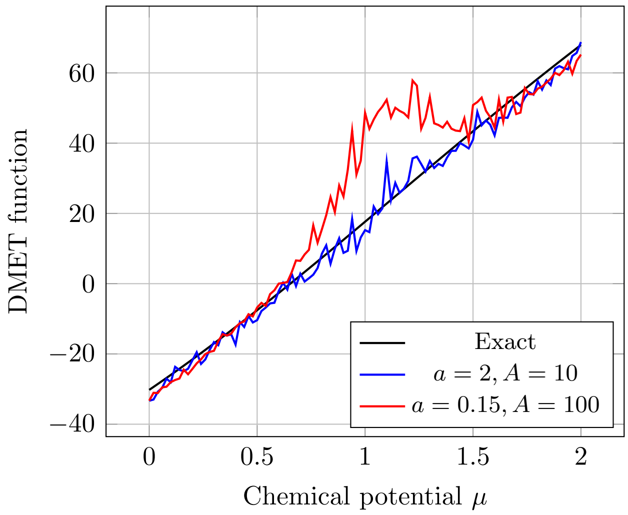


Figure 5.11: Sweeping through the chemical potential for 100 values between $\mu = 0$ and 2, using different values for SPSA metaparameters. At each value of μ, the embedded Hamiltonian is solved and the DMET function is calculated. The graph shows the $U = 4$ 1D Hubbard model with $\langle n \rangle = 0.5\,(N_{\text{occ}} = 120)$ and a fragment size of one. The VQE solver uses depth two HV-min as the ansatz circuit. Note that although the DMET function is a straight line here, this is not always the case.

When we previously used SPSA with the Hubbard model in Chapter 3, we fixed the Hubbard model parameters t and U and tested with different grid sizes. Here, there are many more variables. For example, to produce the curve for fragment size one in Figure 5.10, we fix U, but α and β change with every value of $\langle n \rangle$, and μ changes several times for each H_{emb}. One way to check whether SPSA is effective in all of these regimes is to start by fixing U and $\langle n \rangle$ (and thereby α and β), but sweep through μ for a range of values. At each value of μ we use SPSA with VQE to solve H_{emb} and calculate the DMET function (equation (4.13)). The DMET loop terminates when the root of this function is found; if SPSA leads to wrong values of the DMET function (beyond statistical noise), then it will not be suitable for use in DMET.

A plot of this is shown in Figure 5.11 for the $U = 4$ 1D Hubbard model at quarter-filling with a fragment size of one. Each term in the expectation value was estimated using 10^4 samples, and the final state at the end with 10^5. We set the maximum number of SPSA iterations to 2,000 and ran standard SPSA using the metaparameters from Section 3.2.2, and with metaparameters $a = 2, A = 10$ for a higher rate of convergence. A "bump" appears in the DMET function when using the original metaparameters. We observed that this bump reduced in size when the number of SPSA iterations increased. We also observed that the bump moved position for different values of U and $\langle n \rangle$, and in fact was roughly centred on $\beta \approx -\mu$ (note that in Figure 5.11, $\beta = -1.2005$).

A possible reason for this bump is the ansatz we are using, HV-min. In this ansatz, all of the number gates are parametrised with the same angle θ i.e. we implement $e^{-i\mu\theta n_{0/2}}$ and $e^{i\beta\theta n_{1/3}}$. When $\beta = -\mu$, this has the effect of applying a global phase to the state, and so the angle θ contributes nothing to the ansatz and its gradient is zero. When $\beta \approx -\mu$, the gradient in the direction of θ is small and this has the effect of shrinking the magnitudes of random gradient that SPSA takes. It is possible that having a small step size combined with a small gradient means that SPSA cannot move away from the start point effectively enough and it gets stuck. By increasing the SPSA metaparameters which govern the step size by an order of magnitude i.e. $a = 2, A = 10$, we are less likely to get stuck early on in the optimisation routine. Note that the bump is not present when using the HV-max ansatz, lending weight to this reasoning.

This example demonstrates that it is important to tailor the SPSA metaparameters for different problems, even when using a low number of qubits. In addition, problems which are not present when running simulations using exact values of the expectation can appear when measurements, noise or different optimisation routines are involved.

5.3 Summary

Over the past two chapters, we have carried out a detailed study into how single-shot DMET could be used to solve the Hubbard model on a quantum computer with VQE. We have used the form of the embedded Hamiltonian to construct efficient swap networks for implementing the HV ansatz, and measurement schemes for estimating expectation values. These constructions have assumed that we are using the JW encoding and the architecture of the quantum device is fully-connected.

We also conducted numerical simulations up to a fragment size of four (16 qubits) using exact expectation values from the VQE, and up to fragment size two (8 qubits) involving measurements. The VQE algorithm is a nested optimisation loop providing imperfect solutions to the embedded Hamiltonian within the larger DMET optimisation loop. There is a lot of scope for errors to propagate throughout the algorithm, but despite this the simulations showed that DMET with VQE was effective. The errors on the observables were shown to decrease exponentially with the depth of the ansatz, meaning that it is possible to use a lower depth ansatz if a high accuracy is not required.

DMET is an embedding method which can be used to drastically reduce the number of qubits required to find the ground state properties of a given Hamiltonian. However, in the case of the Hubbard model, applying the embedding procedure leads to an embedded Hamiltonian with a higher complexity than would be obtained by truncating the original one.

For example, let us consider a quantum computer with 64 qubits. We could solve a 4×8 Hubbard model with open boundary conditions using a two-qubit gate depth of 9 per ansatz layer and 5 circuit preparations (see Chapter 3). However, consider a DMET calculation with 16 sites in the fragment; taking the shape of the fragment to be 4×4, one layer of the ansatz would require a two-qubit gate depth of 30, and 32 preparations of the quantum circuit would be needed to measure all of the expectation terms.

This suggests that directly solving the Hubbard model is a more viable goal for very near-term quantum computers, which contain qubits that are quite noisy (ruling out high-depth circuits). Although there have been several small-scale demonstrations of DMET on quantum hardware [140, 141], larger demonstrations would likely benefit from slightly higher quality qubits, capable of running higher depth circuits. The use of DMET in this latter regime could allow the simulation of molecules using a greatly reduced number of qubits, compared to the thousands of qubits that a direct simulation could require.

Chapter 6

Simulating quantum computing

It is important to have fast, high-performance quantum computing simulators to be able to fully investigate NISQ algorithms. An algorithm such as the VQE can be costly to simulate due to the large number of circuits that need to be run. For example, in one of our VQE simulations solving the Hubbard model on 3×4 grid, the depth 10 EHV ansatz took almost 48 hours to run on a GPU. A single circuit run takes around 7 seconds – which is fairly fast for a 24-qubit circuit with thousands of gates – but the optimisation routine required us to run 24,000 different circuits [58]. This run time would only increase if we wanted to simulate measurements or noise, thereby limiting the size of practical VQE simulations.

To code a fast quantum simulator, we must properly utilise a fast compiled language such as C or C++ whilst also implementing the most efficient algorithms for simulating gates and measurements. Many quantum simulators, for example ProjectQ [148], Qiskit [149] and Cirq [150], have a backend coded in C or C++ that is accessed through a Python interface. Although the backend is often very fast and well optimised, the overhead of the communication between C/C++ and Python can make simulations of VQE slow. This is one of the reasons why QuEST [56], a simulator written in C and fully accessible through C/C++, was used in the VQE and DMET simulations done in Chapters 3 and 5[1].

This chapter is the result of efforts to further speed up these simulations of VQE by improving the performance of quantum computing simulators for small numbers of qubits. We present a new quantum simulator coded in C++ called the Quantum Simulation Library (QSL) [61] that we use to test out different ideas. Table 6.1 compares the current features of QSL with a selection of other quantum simulators

---

[1] An initial prototype of the VQE code written in C++ and using QuEST was almost 10,000× faster than the prototype written in Python and using ProjectQ.

Simulator	C/C++ access	GPU access	SSE or AVX	Sampling	Gate fusion	Number pres.
QSL	✓			✓		✓
QuEST [56]	✓	✓				
ProjectQ [148]			✓		✓	
Qiskit (Aer) [149]		✓	✓	✓	✓	
Cirq (qsim) [150, 155]		✓	✓	✓	✓	✓
Qulacs [156]	✓	✓				
Intel-QS [157, 158]	✓		✓		✓	
Quantum++ [159]	✓					

Table 6.1: A non-exhaustive list of quantum computing simulators partially or fully coded in C/C++. The criterion for whether a simulator is accessible from C/C++ is ease of use; for example, to use Qiskit Aer from C++ its CMake file must be modified, and although qsim has a C++ interface, it is difficult to use due to a lack of documentation. SSE and AVX are advanced vectorised instructions that exploit data parallelism. Sampling refers to whether the simulator can rapidly simulate measurements by sampling from the state vector (see Section 6.3). Gate fusion is the process of combining gates together before applying to the state vector. Optimisations relating to number preservation are discussed in Section 6.4.

that are partially or fully coded in C or C++; see [151] for a more exhaustive list of simulators. Tensor network methods allow for the simulation of a larger number of qubits (35+) with a low circuit depth [14, 15, 152, 153], but are slower for smaller simulations due to the overhead they introduce. We therefore restrict the discussion in this chapter to the simpler method of simulation where the entire state vector is stored in memory; this is known as Schrödinger-style simulation [154].

We start in Section 6.1 with a description of how the state vector is stored and its memory requirements. In Section 6.2 we discuss different methods for simulating one- and two-qubit gates, and compare our QSL gate implementations with QuEST, finding that our gates can be up to 8-10× faster. We then describe the important subject of simulating measurements, in Section 6.3. In particular, we present the details of the measurement sampling scheme mentioned in Chapters 3 and 5. In all of our VQE and DMET simulations, the ansatz circuit conserved the fermion occupation number. We conclude this chapter by showing how this circuit property could be used to speed up simulations in Section 6.4. This optimisation can be applied to other quantum chemistry problems when the quantum circuit is number-preserving. The work in this chapter is based on unpublished work done jointly with John Scott whilst coding QSL [61].

6.1 Storing the state vector

To fully represent an arbitrary n-qubit state

$$|\psi\rangle = \sum_{i=0}^{2^n-1} \psi_i |i\rangle \tag{6.1}$$

on a classical computer, we must store the 2^n complex amplitudes ψ_i. A single-precision floating-point number requires 32 bits (4 bytes) of storage, therefore each complex amplitude requires 8 bytes. For a double-precision floating-point number, this goes up to 16 bytes as each individual number requires 64 bits (8 bytes) of storage. Due to the exponential nature of the state vector, each additional qubit doubles the amount of memory required; this quickly becomes too large to store. Table 6.2 shows some examples of the amount of memory required to store the entire state vector in memory.

The number of qubits that can be simulated is limited by the amount of RAM available. Nowadays, laptops come with 8GB of RAM as standard, but not all of this is available for use as some of it is required by the operating system and other programs. Therefore, with a standard laptop it is possible to simulate at most 28 qubits using double-precision or 29 with single-precision (using 4GB of RAM). It is possible to use part of the hard drive to store the state vector, but this would drastically slow down the simulation due to slower memory read/write speeds [160].

There are many choices that can be made about how to store the state vector. For example, in C++ we could store a `std::vector` of length 2^n with `std::complex` elements, or a length 2^{2n} `std::vector` where the real and imaginary amplitudes are interleaved, or even two separate vectors each storing either the real or imaginary amplitudes. How the state vector is stored will have an effect on the speed of the computation. In fact, when applying simple gates such as X and CNOT it is advantageous to have the real and imaginary amplitudes next to each other in memory. This is because the operation of these gates requires swapping amplitudes, meaning that the real and imaginary parts can be swapped as a pair (see Section 6.2). However, for more complicated gates involving complex multiplication, the three different ways of storing the state become comparable. We choose to implement the first method in QSL as it is conceptually simple. The only change that we make is to use a user-defined `struct` for the complex numbers instead of `std::complex`.

Number of qubits	Single-precision	Double-precision
8	2 KB	4 KB
16	512 KB	1 MB
24	128 MB	256 MB
32	32 GB	64 GB

Table 6.2: Amount of memory required to store the state vector for some representative qubit numbers. Note that 1 KB = 1024 bytes.

Another choice that can be made is about the "endianness" of the vector indices. In a big-endian system, the most significant bit represents the lowest-numbered qubit. For example, if the state is $|01\rangle$ then qubit 0 is in state $|0\rangle$ and qubit 1 in state $|1\rangle$. In a little-endian system, the least significant bit represents the lowest-numbered qubit. Here qubit 0 is in state $|1\rangle$ and qubit 1 in state $|0\rangle$. The two systems can become easily confused; big-endian is more natural for our left-to-right writing system but little-endian is mathematically simpler. For example, the index of the computational basis state with qubit k in state $|1\rangle$ and the other qubits in state $|0\rangle$ is indexed by 2^k in little-endian, but 2^{n-k-1} in big-endian. Therefore, we make the choice of using the little-endian system in QSL, meaning that the index of any computational basis state can be calculated as follows,

$$|x_{n-1}\, x_{n-2}\, \dots\, x_1\, x_0\rangle \mapsto \sum_{k=0}^{n-1} 2^k x_k, \quad x_k \in \{0,1\}. \tag{6.2}$$

We finish this section by briefly discussing a naïve way of implementing gates. Since multi-qubit states are constructed through the use of the tensor product, gates can also be constructed in this way. If we wanted to apply a one-qubit gate to a specific qubit, we could pad the gate with identities on either side to create a $2^n \times 2^n$ matrix and then multiply this with the state vector. An example on four qubits with the X gate applied to qubit 1 is:

$$I \otimes I \otimes X \otimes I \begin{pmatrix} \psi_{0000} \\ \psi_{0001} \\ \vdots \\ \psi_{1110} \\ \psi_{1111} \end{pmatrix} = \begin{pmatrix} \psi_{0010} \\ \psi_{0011} \\ \vdots \\ \psi_{1100} \\ \psi_{1101} \end{pmatrix}. \tag{6.3}$$

Although this $2^n \times 2^n$ matrix is sparse, the number of non-zero elements scales exponentially with the number of qubits and at best is equal to the length of the state vector. Since storing the state vector is already a memory intensive task, it is best not to use any additional memory beyond what is required. This is one of the reasons why gate operations are actually carried out through the use of in-place matrix multiplication. For example, in the case of the X gate, this only requires the storage of the 2×2 matrix representing the X gate. In the next section we discuss how this is done in practice using bitstring construction and manipulation.

6.2 Simulating one- and two-qubit gates

In this section we discuss efficient implementations of one- and two-qubit gates. We compare our gates in QSL with QuEST to show that the simulations carried out in this thesis could be sped up.

6.2.1 One-qubit gates

To apply an arbitrary one-qubit gate $U = \left(\begin{smallmatrix}\alpha & \beta \\ \gamma & \delta\end{smallmatrix}\right)$ to qubit k, the elements of the state vector are modified as follows:

$$\begin{pmatrix}\psi_i \\ \psi_{i+2^k}\end{pmatrix} \mapsto \begin{pmatrix}\alpha & \beta \\ \gamma & \delta\end{pmatrix} \begin{pmatrix}\psi_i \\ \psi_{i+2^k}\end{pmatrix}, \tag{6.4}$$

where i is every bitstring of length n with 0 in the kth position. Since there are 2^{n-1} such i, this corresponds to a loop that is half the length of the state vector. The body of the loop contains four complex multiplications and two addition, corresponding to equation (6.4). The way in which the indices i are calculated and how the matrix multiplications are carried out within the loop are the two key components of simulating one-qubit gates; the ultimate aim is to reduce the number of operations done inside the loop. In this section we discuss both of these components, starting with the indexing.

In QuEST, the index i is calculated using the formula $i = \lfloor j/2^k \rfloor 2^{k+1} + (j \bmod 2^k)$ for $j = 0, \dots, 2^{n-1} - 1$ [56]. This requires two integer divisions, a multiplication and an addition for each value of j. Whereas multiplications and additions are extremely fast operations on the processor, integer division is substantially slower[2] [161]. Another way of calculating the index i is to insert a bit between the kth and $(k+1)^{\text{th}}$ bit of j. Pseudocode for how this can be achieved with bitstring manipulation is as follows:

---

[2] For example, on the Intel Skylake-X processor, the `IDIV` (integer division) instruction may be between 90-360 times slower than the `ADD` instruction.

```
lower_mask = (1 << k) - 1
upper_mask = ~lower_mask
i = (j & lower_mask) + ((j & upper_mask) << 1)
```

The `&` is bitwise AND, `~` is bitwise NOT and `<<` is the operator which shifts bits to the left, i.e. `1 << k` is equivalent to 2^k. The masks can be calculated outside of the loop; bit-shifting and bitwise AND are of almost comparable speed to addition [161] making this method of calculating the indices faster than the previous one. An example of a quantum simulator which uses this method is Qulacs [156].

In QSL we use yet another method which only requires one addition to calculate i. Two nested `for` loops are used, one which loops through all possible bitstring combinations below bit k, and one which loops through all bitstring combinations for the bits above k. This is demonstrated along with the application of the gate U in Algorithm 1. An example of a quantum simulator which also uses this method is Intel-QS [157, 158].

Algorithm 1 Apply an arbitrary one-qubit gate to qubit k

```
for $s \leftarrow 0$ to $2^n - 1$ in steps of $2^{k+1}$ do        ▷ Loop through upper bits
    for $r \leftarrow 0$ to $2^k - 1$ do                          ▷ Loop through lower bits
        $i_0 \leftarrow s + r$                    ▷ Calculate the indices with 0 and 1 in position $k$
        $i_1 \leftarrow i_0 + 2^k$
        $\psi_{i_0} \leftarrow \alpha\psi_{i_0} + \beta\psi_{i_1}$          ▷ Update the state vector amplitudes
        $\psi_{i_1} \leftarrow \gamma\psi_{i_0} + \delta\psi_{i_1}$
    end for
end for
```

We can now discuss how the matrix multiplications can be carried out within the loop. We can optimise the updating of the state vector amplitudes for common gates by hard-coding the specific operation. Taking the X gate as an example, we only need to swap ψ_{i_0} and ψ_{i_1}, completely getting rid of any floating point operations. For the Z gate and when applying a phase shift, we only need to modify ψ_{i_1}, leaving half of the state vector untouched. Even gates such as the Hadamard and rotation gates can be optimised; since their matrix elements are not fully complex, half of the multiplications can be removed. For example, it is not necessary to do full complex multiplication if some real or imaginary components are known to be zero. The tailoring of gate operations is the approach that QuEST takes and that we take in QSL. Many other quantum simulators such as Qiskit [149], qsim [155] and Qulacs [156] do not tailor gate operations, but store the full complex gate matrices to allow for "fusing" operations that combine gates.

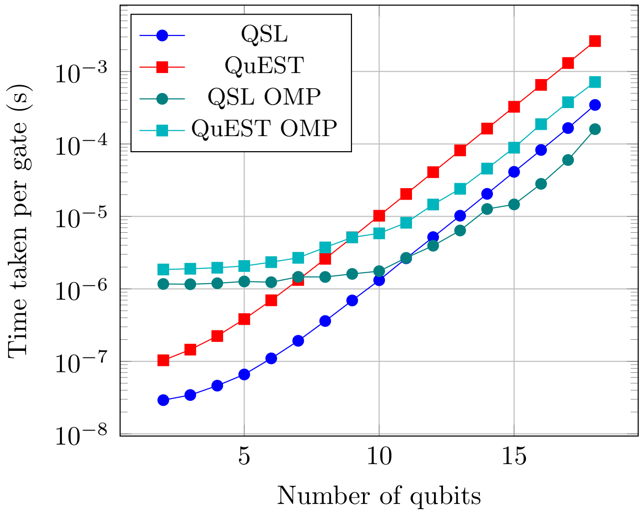


Figure 6.1: Timing of the X-rotation gate in QSL and QuEST, with and without parallelisation. A random n-qubit state vector was generated and the $R_X(\theta)$ gate (with a random angle θ) was applied to each qubit in order. The application of each round of gates was repeated 2^{20-n} times to build up an average. For OMP, 8 threads were used with QuEST and 4 with QSL.

Figure 6.1 compares the time to carries out a representative one-qubit gate, $R_X(\theta)$, using QSL and QuEST. Recall that the main difference between these two simulators is the way in which the state vector indices are calculated. For this specific gate, an additional difference is that QuEST carries out full complex matrix multiplication in the inner loop, whereas in QSL we have only done the necessary multiplications. The resulting gate in QSL is on average 7-8× faster than QuEST. The non-parallelised version of QSL is also faster than the version of QuEST using OpenMP (OMP) with 8 threads. Note that below around 10 qubits OMP introduces an overhead, this is the case for all quantum simulators. Below this threshold, the state vector is small enough that setting up threading, splitting up the loops over the threads and then recombining is slower than just iterating over the state vector.

6.2.2 Two-qubit gates

Controlled two-qubit gates are a natural extension of one-qubit gates. For an arbitrary controlled-U gate, we apply U to the target qubit if the control qubit is in the state $|1\rangle$. Only half of the amplitudes in the state vector need to be modified as opposed to the full state vector for general one-qubit gates. We can apply a controlled two-qubit gate using Algorithm 1 (or any of the other indexing methods described above), but with an additional step in the inner loop checking whether the control

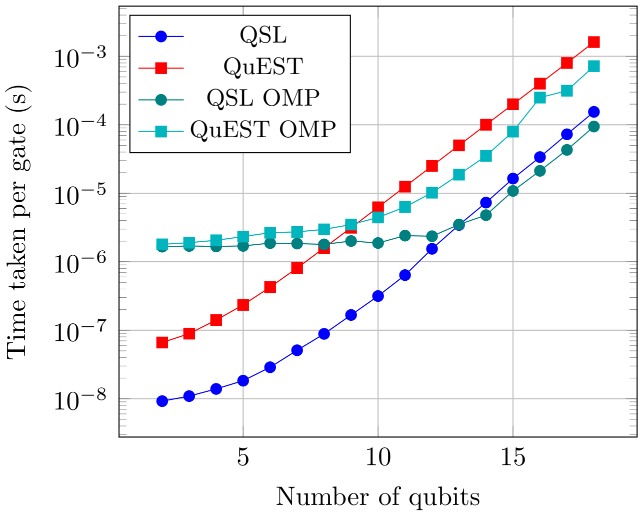


Figure 6.2: Timing of the CNOT gate in QSL and QuEST, with and without parallelisation. A random n-qubit state vector was generated and the CNOT gate was applied between every pair of qubits. The application of each round of gates was repeated 2^{20-n} times to build up an average. For OMP, 8 threads were used with QuEST and 4 with QSL.

bit c is 1 before doing the matrix multiplication on the target qubit t. This check requires a masking operation `j & (1 << c)`, followed by an `if` statement to see if the result is zero or not. This is the method that is implemented in QuEST [56].

A more efficient method (which also works for general two-qubit gates) is to only loop over the indices where the state vector will need to be updated. Similarly to how the bitstrings for the one-qubit gates were constructed in two parts and the kth bit inserted, the bitstrings for two-qubit gates can be constructed in three parts: the string below $\min(c, t)$, the string between c and t, and the string above $\max(c, t)$. We can do this either by looping from $j = 0, \ldots, 2^{n-2} - 1$ and breaking up this bitstring over the three parts, or by using three nested `for` loops. In QSL we use the second method.

A disadvantage of this multi-loop method is that it does not generalise well to multi-qubit gates. For example, if we wanted to do an m-qubit gate, then we would require $m + 1$ nested `for` loops; it is not possible to dynamically generate `for` loops so the code would become unreadable. A combination of checking bits and breaking up bitstrings over multiple parts could be used for multi-qubit gates, but for now QSL is restricted to one- and two-qubit gates.

Figure 6.2 compares the time to carry out a representative two-qubit gate, CNOT, using QSL and QuEST. Here, the difference is even larger than the one-qubit gate as QSL is 10× faster than QuEST. The gap between parallelised QuEST and the non-parallelised version of QSL is also much larger than for the one-qubit gate. Note also that the time taken per gate is lower for CNOT than $R_X(\theta)$. This is to be expected since we are modifying only half of the state vector compared to the whole state vector. This has consequences for the way in which circuits are implemented on the simulator. Taking the combined FSWAP and Hopping gate in equation (2.19) as an example, even though on an actual quantum computer it will be faster to implement two one-qubit gates $S^\dagger$ followed by the Hopping gate, for simulation purposes it will be advantageous perform the FSWAP directly.

6.3 Simulating measurements

Information is extracted from quantum computers through the measurement of qubits; a quantum simulator needs to be able to simulate realistic measurements (not, for example, by just adding Gaussian noise to expectation values) to be able to study the behaviour of quantum algorithms more accurately. Simulating the measurement and collapse of a single qubit can be implemented by looping over the state vector twice and generating a random number; this procedure is given in Algorithm 2.

Algorithm 2 Measure and collapse qubit k

```
$p \leftarrow \sum_{x:x_k=0} |\psi_x|^2$        ▷ Calculate the probability of measuring qubit $k$ as 0
$r \leftarrow \mathrm{rand}(0,1)$
if $r < p$ then                                  ▷ Generate random measurement outcome
    $m \leftarrow 0,\ f \leftarrow \sqrt{p}$
else
    $m \leftarrow 1,\ f \leftarrow \sqrt{1-p}$
end if
$\psi_{x:x_k \neq m} \leftarrow 0$               ▷ Zero out amplitudes for the opposite outcome
$\psi_{x:x_k=m} \leftarrow \psi_{x:x_k=m}/f$     ▷ Re-normalise the rest of the state vector
```

Unless we are simulating an algorithm that involves measuring out qubits as an intermediate step, using this procedure to simulate measurements is inefficient. For an algorithm such as the VQE, taking an estimate of the energy requires thousands of measurements of the state vector. Using Algorithm 2, we would have to measure and collapse each qubit in turn (requiring $2n$ passes of the state vector), and then re-prepare the state vector for the next measurement, thousands of times.

The simulation of measurement can be sped up using two simple observations. The first is that measuring all of the qubits at once is the same as measuring them one by one – this means that we can write down the overall probability distribution for all of the qubits and sample from that to generate measurement outcomes. The second is that it is unnecessary to simulate the collapse of the state vector when measuring all of the qubits together, as the state vector would collapse to a computational basis state. QuEST implements measurement using the method in Algorithm 2. To speed up our simulations that incorporated measurements, we implemented sampling using these two observations in the C++ programs for simulating VQE and DMET (see Sections 3.2.2 and 5.2.2). In the remainder of this section, we will explain how this was done and compare two different sampling methods.

To sample from the state vector, we first need to compute the vector C of cumulative probabilities

$$C_i = C_{i-1} + |\psi_i|^2, \tag{6.5}$$

where $C_{-1} = 0$ and $i = 0, \ldots, 2^n - 1$. To generate a measurement outcome, we generate a random number r between 0 and 1. If r is in the range $[C_{m-1}, C_m)$ then m is the measurement outcome. Finding this range is a search problem; a naïve way of finding it is to perform a linear search, but in the worst case scenario this would require iterating through the whole 2^n length vector C. In the VQE and DMET code we use a binary search algorithm, which runs in logarithmic time, to determine the range and hence the measurement outcome.

Using this method, the cumulative probability distribution is only generated once and then used repeatedly to generate samples. The drawing of one sample from the distribution corresponds to generating a random number and performing a binary search. It is also possible to reduce the length of C by just storing the outcomes that have a non-zero probability of occurring; this can be done by taking into account problem properties such as number preservation (see Section 6.4) or simply checking if $|\psi_i|^2$ is zero.

An alternative sampling method, which is used in qsim [155], is to generate all of the random numbers at once, sort them into ascending order and then iterate through C binning the elements of the sorted random vector into the ranges $[C_{m-1}, C_m)$. If we wish to generate M samples, we must store a vector of random numbers of length M. Unlike the binary search method, we do not need to store C as we only need to maintain the running cumulative probability for the purpose of binning the random numbers.

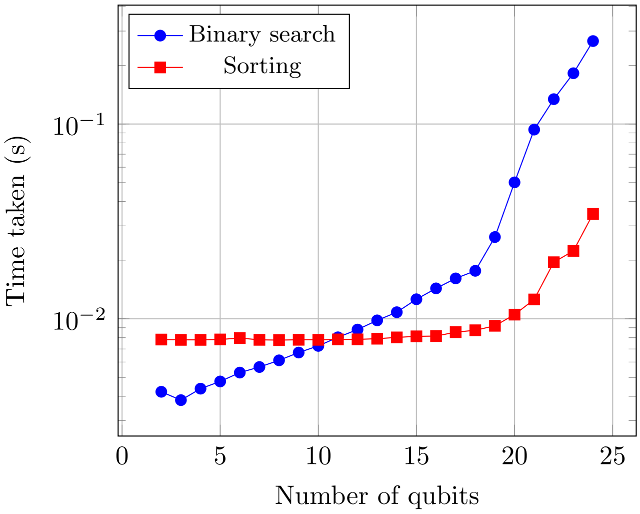


Figure 6.3: Comparison of the binary search and sorting methods of measurement sampling in QSL for $M = 100,000$ and $n = 2$ to 24. A random n-qubit state vector was generated and M measurement samples were taken. The prediction for when the two methods crossover is at $n = 16$.

We can perform a comparison of the time complexities of both of the methods when taking M samples from an n-qubit state vector. Both methods have an overhead of $O(2^n)$ for preparing or iterating through C. The binary search method requires us to search through a vector of length 2^n, M times; this has a complexity of $O(Mn)$. The sorting method requires us to sort a random vector of length M. Using the quicksort algorithm this has a complexity of $O(M \log M)$ [162]. A rough theoretical calculation predicts that the binary search method will be faster if $n \lesssim \log M$. We coded both methods in QSL to test this prediction and found that in practice the point at which the two methods crossover is lower. For example, for $M = 1,000$ the sorting method is faster beyond $n = 7$ compared to the prediction of $n = 9$. For $M = 10,000$ the theoretical crossover is at $n = 13$, but we observed it at $n = 9$.

Figure 6.3 demonstrates the performance of the two sampling methods for $M = 100,000$. The sorting method takes an almost constant amount of time until 18 qubits, where the cost of iterating through the state vector to calculate the cumulative probabilities starts to dominate the time. At 24 qubits, the sorting method is almost an order of magnitude faster than the binary search method. However, having both methods available for use in a quantum simulator will be advantageous as the best sampling method can be chosen for the problem at hand. It may be possible to achieve further speed-ups by implementing more advanced sampling techniques such as the alias method, which can draw a sample in constant time [163].

6.4 Number-preserving simulator

All of the circuits that have been used to produce the results in this thesis have had the property that they conserve the fermionic occupation number. In other words, at any point in the circuit, the state vector is a linear combination of computational basis states where the associated bitstrings have the same Hamming weight. Many Hamiltonians relevant to quantum chemistry preserve fermion number, spin, or demonstrate certain symmetries; these properties are routinely used in simulations on classical computers to reduce the memory required to solve problems or speed up calculations. In this section we will show how we can also use these properties to speed up the simulation of quantum circuits, with a focus on number preservation.

For an n-qubit state where the Hamming weight m is preserved, only $\binom{n}{m}$ of the amplitudes in the state vector can be non-zero. Only storing or only operating on these non-zero amplitudes could significantly speed up circuit simulations. There are a growing number of works taking advantage of this. Google's fermionic quantum emulator (FQE) [164] is tailored to quantum circuit simulations that only involve evolutions according to fermionic operators. Number and spin preservation is used to only store the relevant sectors of the state vector. In some cases they find that the FQE, which is written in Python, is faster than qsim, a highly optimised simulator written in C++. More generally, Jaques and Häner [165] present simulations where the state vector is stored in key-value pairs (i, ψ_i), where $\psi_i \neq 0$, regardless of the problem structure. When the state vector is sparse this allows for the simulation of large numbers of qubits.

In QSL, we have coded a number-preserving (NPr) simulator where gates are applied by modifying the $\binom{n}{m}$ non-zero amplitudes. However, we still store the entire 2^n length state vector, and unlike the FQE we only allow for preserving a single Hamming weight m. The NPr simulator restricts the gates that can be applied. Arbitrary one- and two-qubit NPr gates are

$$U_1 = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{pmatrix} \quad \text{and} \quad U_2 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \alpha & \beta & 0 \\ 0 & \gamma & \delta & 0 \\ 0 & 0 & 0 & e^{i\theta} \end{pmatrix}, \tag{6.6}$$

where $U = \left(\begin{smallmatrix} \alpha & \beta \\ \gamma & \delta \end{smallmatrix}\right)$ is an arbitrary one-qubit gate acting on the $\{|01\rangle, |10\rangle\}$ subspace.

To apply U_1 to qubit k, we cannot use the nested `for` loop method described in Algorithm 1 since there is no straightforward way of looping through the desired bitstrings. We need to apply the phase $e^{i\theta}$ to all amplitudes in the state vector where the index is a bitstring of length n with a Hamming weight of m, and a 1 in the kth position. To do this, we generate base bitstrings of length $n - 1$ with Hamming weight $m - 1$ and then insert a 1 between bits k and $k + 1$. The bit insertion can be done using the pseudocode at the beginning of Section 6.2.1.

A function $f(x)$ which takes a bitstring x with a certain Hamming weight and returns the next (numerically ordered) bitstring y with the same Hamming weight is as follows [166]:

```
t = x | (x - 1)
y = (t + 1) | (((~t & -~t) - 1) >> (ctz(x) + 1)).
```

The `|` is bitwise OR and `ctz(x)` counts the number of trailing zeros in x i.e. the number of bits below the first (least significant) occurrence of 1. In the GNU C compilers it is a built-in compiler intrinsic `__builtin_ctz`.

The simplest way to obtain the indices to implement U_1 is to advance through all base bitstrings using $f(x)$[3], and at each stage insert a 1 between the bits k and $k + 1$. However, due to the cost of computing $f(x)$, we found that in some cases (for example, when m was close to $n/2$) this on-the-fly method was slower than the standard QSL simulator. An improved method is obtained by realising that the base bitstrings do not depend on the index k. As as result, it is possible to pre-compute the base bitstrings and store them in a lookup table. This method results in a speedup because it is no longer necessary to compute $f(x)$ multiple times inside each gate. It allows for parallelisation, unlike the on-the-fly generation which is a serial process. Note that in QSL our lookup table is a `std::vector` of length $\binom{n-1}{m-1}$.

Figure 6.4 demonstrates the speed of the U_1 gate using the standard and NPr (with lookup table) simulators in QSL for 16 qubits. The curves for the NPr simulator are clearly shaped like the binomial distribution, following the length of the lookup table. The parallelised NPr simulator has an overhead for small (and large) values of m as the loop length is small enough that setting up threading dominates the run time. For $m = n/2$ (i.e. half-filling for fermionic systems) we might expect a $2^{15}/\binom{15}{7} \approx 5.1\times$ speedup over the standard simulator, but we find approximately

---

[3] The starting bitstring has a length of $n - 1$ with 1 in the least significant $m - 1$ positions. We end at the length $n - 1$ bitstring with 1 in the most significant $m - 1$ positions.

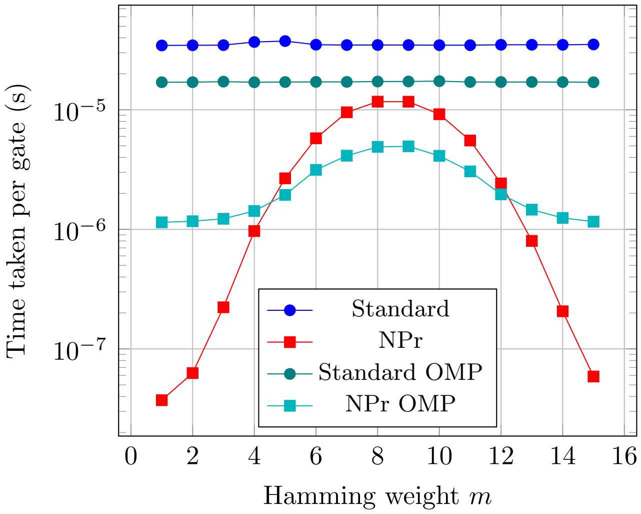


Figure 6.4: Comparing the standard and NPr simulators in QSL using the phase shift gate U_1. A random 16-qubit state vector was generated for Hamming weights of 1 to 15, the phase gate (with a random angle θ) was then applied to each qubit 1000 times to build up an average. For OMP, 4 threads were used.

a 2.1× and 3.5× speedup with and without OMP respectively. Not reaching the theoretical speedup could be due to additional memory access times.

We can extend this procedure of generating base bitstrings of a specific Hamming weight to applying two-qubit gates on qubits k_1 and k_2. Base bitstrings of length $n-2$ and the appropriate Hamming weight are generated, followed by bit insertions between bits k_1 and k_1+1, and bits k_2 and k_2+1. To apply U_2, we now require two lookup tables; one to keep track of the base bitstrings required to apply U on the $\{|01\rangle, |10\rangle\}$ subspace, and one to apply the phase $e^{i\theta}$ to the $|11\rangle$ subspace. Table 6.3 contains a list of the lookup tables required to implement all one- and two-qubit NPr gates. Five lookup tables are needed in total and each is a `std::vector` of size $\binom{\text{Bitstring length}}{\text{Hamming weight}}$. Although additional memory is required to store these lookup tables, in QSL the focus is on speed for a low number of qubits. Furthermore, the length of all the lookup tables combined does not add up to the length of the state vector; for example for $n = 16$ and $m = 8$, the combined length of the lookup tables is 32,175 which is approximately half the length of the state vector.

The work presented in this section could also be extended to spin preservation. Half of the qubits could be designated as spin-up and the other half as spin-down. The number of non-zero elements in the state vector would then be $\binom{n/2}{m_1}\binom{n/2}{m_2}$, where m_1 and m_2 are the fermion occupation numbers/Hamming weights for the two spin-types. Taking 16 qubits at half-filling with an even split of spins, the

Operation type	Bitstring length	Hamming weight
Indexing entire state vector	n	m
One-qubit gate on $\|0\rangle$	$n-1$	m
One-qubit gate on $\|1\rangle$	$n-1$	$m-1$
Two-qubit gate on $\{\|01\rangle, \|10\rangle\}$	$n-2$	$m-1$
Two-qubit gate on $\|11\rangle$	$n-2$	$m-2$

Table 6.3: List of lookup tables required for all one- and two-qubit NPr gates, split by the subspace the gates act on. The lookup tables for the one-qubit gates are also used for collapsing the state vector when a measurement is done. The lookup table in the top row is used to generate the cumulative probability distribution for measurement sampling. The length of each lookup table is the bitstring length choose Hamming weight.

standard simulator could operate on all $2^{16} = 65,536$ elements of the state vector, whereas the NPr simulator would act on at most $\binom{16}{8} = 12,870$ elements (saving a factor of 5), and for a spin-preserving simulator this would drop to $\binom{8}{4}^2 = 4900$ elements (saving a factor of 13). The spin-preserving simulator would be slightly more complex than the NPr one. For example, we would have to restrict the gates that could be applied between the different spin sectors, and loops over the state vector would need to be constructed from two lookup tables (one for spin-up and one for spin-down).

6.5 Summary

In this chapter we have explored practical aspects relating to the simulation of quantum computers, with a focus on the fast simulation of small numbers of qubits. By choosing optimal algorithms we can significantly speed up simulations of quantum circuits. We introduced QSL, a simulation library written in C++, based on the ideas presented in this chapter. Our one- and two-qubit gates were found to be around 8-10× faster than QuEST, the simulation library that was used to produce the results in this thesis. We demonstrated efficient methods for measurement sampling, which speeds up ideal quantum computer simulations of VQE.

We also wrote an NPr simulator in QSL, which takes advantage of the structure of the state vector to speed up circuits for quantum chemistry problems. This NPr simulator had a further 2-3× speedup in the worst case scenario and several orders of magnitude in the best case over standard QSL. In fact, when running the DMET code for $N_{\text{frag}} = 4$ ($n = 16, m = 8$) with the depth two HV-min ansatz, the code on a GPU with QuEST took 17 seconds to run, but only 13.6 seconds with the NPr OMP simulator on a standard laptop. This shows that it is not always necessary to use specialist or high-performance hardware to speed up simulations; simply using the best algorithms and tailoring to the problem at hand can also have a big effect.

We conclude by stressing the importance of writing simulators, and frameworks for NISQ algorithms, in fast compiled languages such as C and C++. The large scale simulations done in this thesis would not have been possible without the use of a language such as C++. Although many quantum simulators written in Python have a fast backend coded in C/C++ [148–150], ultimately the communication between the two programming languages slows down simulations of small numbers of qubits by several orders of magnitude compared to pure C/C++ [56]. This Python overhead may not be prohibitive for single circuit runs, but for an algorithm such as the VQE which requires thousands of circuit runs, it can restrict the number or size of simulations that can be carried out in practice. By coding purely in C/C++, we are able to quickly prototype different ansätze and optimisation algorithms for the VQE, thereby pushing forward research into NISQ algorithms.

Chapter 7

Conclusion

In this thesis, we have carried out an in-depth investigation into how the Hubbard model could be solved using the VQE algorithm on NISQ devices. We considered the solution of the model directly in Chapter 3, and a compressed form obtained by DMET in Chapter 5. We took two main approaches in performing this analysis; calculating the resources required to run the ansatz circuits on ideal quantum computers, and carrying out high-performance classical simulations of these ansatz circuits.

In the first approach, we designed efficient ansatz circuits based on fermionic swap networks to obtain detailed estimates of circuit depths. We also introduced the notion of non-crossing measurements to minimise the number of circuit preparations required to measure expectation values. While the Hubbard model is an important benchmark system in its own right, its simple structure facilitates an easier implementation of VQE than for typical electronic structure Hamiltonians. An important direction for future work is to carry out a similarly detailed analysis of the complexity of VQE for other practically relevant electronic systems.

For the other approach, we conducted numerical simulations of VQE up to twelve sites (24 qubits) for the Hubbard model and four fragment sites (16 qubits) for DMET. These are amongst the largest VQE simulations that have been carried out. However, we are beginning to reach the limit of practical VQE simulations. For example, for the largest Hubbard model grid sizes, simulating high-depth ansatz circuits took days of run time; this is despite running C++ code on a GPU. Quantum computing simulators which take advantage of known properties of certain quantum chemistry problems, such as the number-preserving simulator introduced in Section 6.4 and Google's fermionic quantum emulator [164], could lead to much faster simulations. Increasing the speed of the VQE simulations will mean being able to test more ansätze at larger grid sizes, and using a wider variety of Hubbard model parameters. It will also mean that simulations involving measurements, that were restricted to 18 qubits for the Hubbard model and eight qubits for DMET, could be extended to larger numbers of qubits. We provided efficient algorithms for the simulation of gates and measurement, that might form the basis of future numerical experiments, in Chapter 6.

While running simulations involving realistic measurements and noise gives us some idea of the challenges involved in running VQE, in particular relating to the classical optimisation, it is not a replacement for performing experiments on an actual quantum device. For example, Montanaro and Stanisic [117] ran the HV ansatz circuit described in Chapter 3 for the 2×1 Hubbard model (four qubits) on a Rigetti Aspen device, and found that due to device noise, it was not possible to find the ground state unless the problem was compressed onto two qubits. A key direction for future work, that we have not explored in detail in this thesis, is the development of error mitigation techniques for VQAs [20] – requiring more realistic noise models and more experiments on quantum devices.

Research into appropriate algorithms and applications for near-term quantum devices are crucial to furthering the field of quantum computing. The Hubbard model is a good target for NISQ devices due to its simple structure and relevance to important problems such as high-temperature superconductors [49]. It is likely that solving the Hubbard model could be one of the first demonstrations of quantum advantage. However, we are still a while away from seeing such demonstrations; for example, running a 53-qubit computation with a two-qubit gate depth of 20 on the Google Sycamore device has a 0.2% fidelity [1]. Gate fidelities would need to be improved by several orders of magnitude to make the gate depth of 325 estimated for solving the 5×5 Hubbard model in Chapter 3 a reality, and implementing methods such as DMET is even further out of reach for today's quantum computers. Ultimately, improvements in quantum hardware will be needed to reach a quantum advantage.

Appendices

Appendix A

The number-preserving ansatz

In this appendix we will go into some detail about the choices that can be made when implementing the NPr ansatz introduced in Section 3.1. As with many ansätze, we must specify properties such as starting parameters and initial states. The work in this appendix is based on Appendix C of the paper "Strategies for solving the Fermi-Hubbard model on near-term quantum computers" [57].

In addition to the use of the ground state of the non-interacting Hubbard model as an initial state, the NPr ansatz also allows a computational basis initial states (with the correct fermionic occupation number). All gates in the circuit preserve the number of fermions, so the VQE method will find the ground state of H_{hub} restricted to the chosen occupation number subspace. This allows a saving in initial complexity compared with starting in the ground state of the non-interacting model (although with an associated penalty in terms of the number of layers required to find the ground state).

The sites we choose to be occupied by fermions can make a significant difference to the complexity at a fixed depth. We ran a number of tests brute forcing all the possible starting states on selected small grid sizes. We found that in many cases the best states reached errors several orders of magnitude better than the worst states, but given the small grid sizes considered, the pattern for picking these good states remains unclear.

An intuitive approach would be to place fermions evenly across the grid, allowing them to quickly spread out. Then the ground state can be produced from the initial state using potentially fewer layers of the ansatz circuit, if it does indeed correspond to a spread-out state. Empirically, we observed that the optimiser performed better with this layout than a naïve one where fermions are placed at the top left corner of the grid, although we note that other schemes might yield even better results.

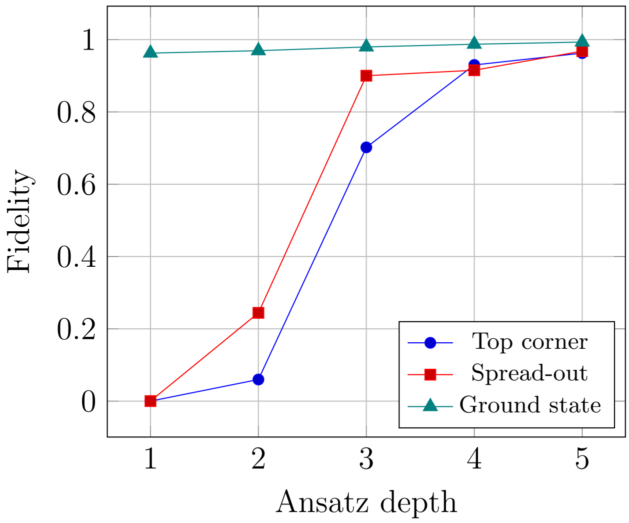


Figure A.1: Comparison of initial fermion placements against using the ground state as a initial state for a 3×3 grid occupied by six fermions. For the spread-out state we placed fermions of both spin-types for the three sites along the main diagonal of the grid. The spread-out placement generally performs better than the top-corner placement that fills the first six orbitals, especially for lower depths. Only starting in the ground state achieves fidelity 0.99, while the others reach around 0.96 in depth five.

Figure A.1 gives a demonstration for a 3×3 grid occupied by six fermions.

For a 3×3 grid, the ground state of the non-interacting model can be prepared in depth 8 (assuming a fully-connected architecture), whereas each NPr ansatz layer requires depth 7. So, in this case, starting with a computational basis state does not seem to be advantageous. We further remark that the NPr ansatz starting from a computational basis state cannot find the true ground state of the non-interacting Hubbard model in the case where the number of fermions with each spin is one. This is because all computational basis states with Hamming weight one are in the null space of this model, and hopping terms preserve this subspace [57].

When starting with a computational basis state, the ansatz (and therefore the optimiser) has to do more work to produce something close to the ground state of the full model (compared with starting in the $U = 0$ ground state). To reduce the work that the optimiser needs to do, we can first find an ansatz circuit that produces a state close to the ground state of the non-interacting model by classically simulating the VQE procedure. Because we only need to consider a single spin, the number of qubits in the simulation is halved. For small grid sizes feasible on near-term quantum devices, the non-interacting problem will be tractable on a classical computer. An

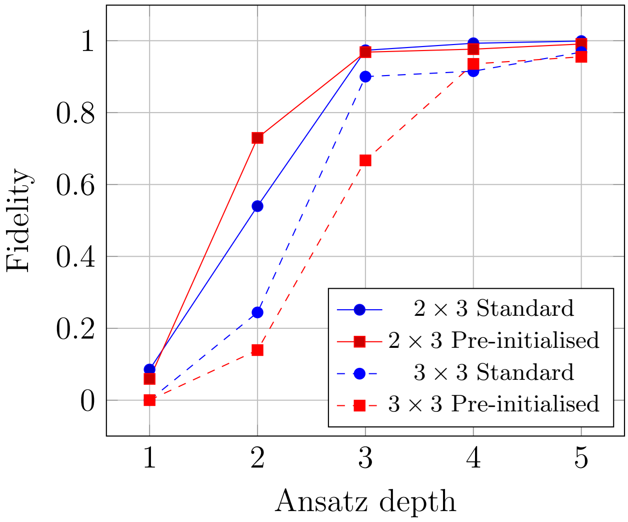


Figure A.2: Comparison of the pre-initialised NPr ansatz and the standard NPr ansatz for 2×3 occupied by four fermions and 3×3 occupied by six fermions. The initial placement of the fermions is spread out (for 2×3 two sites at opposite corners of the grid are fully occupied, 3×3 is explained in Figure A.1). Pre-initialisation improves the results for 2×3 depth two, but makes it worse for 3×3 in all cases. The difference between the ordinary and pre-initialised ansatz reduces as the depth increases; similar behaviour was demonstrated in Figure A.1.

advantage of classically simulating the procedure (rather than also running these smaller instances on a quantum computer) is that we can use exact expectation values.

Once we have performed the optimisation classically, we can pre-initialise the parameters of the full-model ansatz by using the final parameters from the non-interacting model. The intuition is that by allowing the optimisation procedure to begin with a circuit that produces the ground state of the non-interacting model (which we know is a good choice from Figure 3.7), it then 'only' has to optimise this circuit to produce a ground state of the complete model, having already been pointed in the right direction.

However, it is not clear when this procedure is beneficial, for some grid sizes and depths it causes the ansatz to perform worse. Figure A.2 demonstrates this for 2×3 and 3×3 grids where the initial placement of the fermions is spread-out. We note that different placements change how effective the pre-initialised ansatz is, which requires further investigation.

Appendix B

Calculating the 1-RDM of a Slater determinant

In this appendix we will show that the 1-RDM ρ of a Slater determinant is $\rho = \Phi\Phi^\dagger$, where Φ is the matrix representation of a Slater determinant (each column is an occupied orbital written as a linear combination of the original orbitals). This fact is commonly stated but finding a derivation is difficult. The derivation presented here was included in Appendix B of "Solving the Hubbard model using density matrix embedding theory and the variational quantum eigensolver" [59].

Let N be the number of spin-orbitals in a system, M of which are occupied by fermions. An arbitrary Slater determinant $|\Psi\rangle$ can be written as

$$|\Psi\rangle = \prod_{\mu=1}^{M} c_\mu^\dagger |\mathrm{vac}\rangle, \tag{B.1}$$

where $\{c_\mu^\dagger\}_{\mu=1}^{M}$ are the occupied orbitals and $|\mathrm{vac}\rangle$ is the vacuum state. The occupied orbitals can be written in terms of the original spin-orbitals $\{a_k^\dagger\}_{k=1}^{N}$,

$$c_\mu^\dagger = \sum_{k=1}^{N} a_k^\dagger \Phi_{k\mu} \tag{B.2}$$

for $\mu = 1, \ldots, M$, where Φ is an $N \times M$ matrix of coefficients writing the occupied orbitals in terms of the original orbitals [142]. The $\{a_k^\dagger\}$ form a basis for the system, but the $\{c_\mu^\dagger\}$ may not since there are only $M \leq N$ of them. However, they can be expanded to form a basis by adding in $N - M$ more linearly independent $c_\mu^\dagger$ so that equation (B.2) now applies for $\mu = 1, \ldots, N$. A consequence of this is that the original orbitals can now be written in terms of the occupied orbitals as

$$a_k^\dagger = \sum_{\mu=1}^{N} c_\mu^\dagger \Phi_{\mu k}^* \quad \text{and} \quad a_k = \sum_{\mu=1}^{N} c_\mu \Phi_{k\mu}. \tag{B.3}$$

Now calculating the 1-RDM becomes

$$\begin{aligned}
\rho_{ij} &= \langle\Psi|a_j^\dagger a_i|\Psi\rangle \\
&= \langle\Psi|\left(\sum_{\mu=1}^{N} c_\mu^\dagger \Phi_{\mu j}^*\right)\left(\sum_{\nu=1}^{N} c_\nu \Phi_{i\nu}\right)|\Psi\rangle \\
&= \sum_{\mu\nu}^{N} \Phi_{i\nu}\Phi_{\mu j}^* \langle\Psi|c_\mu^\dagger c_\nu|\Psi\rangle \\
&= \sum_{\mu\nu}^{M} \Phi_{i\nu}\Phi_{\mu j}^* = \Phi\Phi^\dagger,
\end{aligned}$$

since

$$\langle\Psi|c_\mu^\dagger c_\nu|\Psi\rangle = \begin{cases} 1, & \text{if } \mu = \nu \text{ and } \mu = 1, \ldots, M \\ 0, & \text{otherwise} \end{cases} \tag{B.4}$$

from the definition of the Slater determinant in equation (B.1).

References

[1] F. Arute et al. "Quantum supremacy using a programmable superconducting processor". *Nature* 574.7779 (2019), pp. 505–510.

[2] Y. Wu et al. "Strong Quantum Computational Advantage Using a Superconducting Quantum Processor". *Physical Review Letters* 127.18 (2021), p. 180501.

[3] P. W. Shor. "Algorithms for quantum computation: discrete logarithms and factoring". *Proceedings 35th Annual Symposium on Foundations of Computer Science*. IEEE Comput. Soc. Press, 1994.

[4] L. K. Grover. "A Fast Quantum Mechanical Algorithm for Database Search". *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*. STOC '96. Philadelphia, Pennsylvania, USA: ACM, 1996.

[5] R. P. Feynman. "Simulating physics with computers". *International Journal of Theoretical Physics* 21.6-7 (1982), pp. 467–488.

[6] S. McArdle, S. Endo, A. Aspuru-Guzik, S. C. Benjamin, and X. Yuan. "Quantum computational chemistry". *Reviews of Modern Physics* 92.1 (2020), p. 01 5003.

[7] Y. Cao et al. "Quantum Chemistry in the Age of Quantum Computing". *Chemical Reviews* 119.19 (2019), pp. 10856–10915.

[8] M. Cerezo et al. "Variational quantum algorithms". *Nature Reviews Physics* 3.9 (2021), pp. 625–644.

[9] K. Bharti et al. "Noisy intermediate-scale quantum algorithms". *Reviews of Modern Physics* 94.1 (2022), p. 015004.

[10] J. Hubbard. "Electron correlations in narrow energy bands". *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences* 276.1365 (1963), pp. 238–257.

[11] "The Hubbard model at half a century". *Nature Physics* 9.9 (2013), pp. 523–523.

[12] M. A. Nielsen and I. L. Chuang. "Quantum Computation and Quantum Information: 10th Anniversary Edition". Cambridge University Press, 2010.

[13] J. Preskill. "Quantum Computing in the NISQ era and beyond". *Quantum* 2 (2018), p. 79.

[14] E. Pednault et al. "Pareto-Efficient Quantum Circuit Simulation Using Tensor Contraction Deferral". arXiv:1710.05867. 2017.

[15] F. Pan and P. Zhang. "Simulating the Sycamore quantum supremacy circuits". arXiv:2103.03074. 2021.

[16] P. Krantz, M. Kjaergaard, F. Yan, T. P. Orlando, S. Gustavsson, and W. D. Oliver. "A quantum engineer's guide to superconducting qubits". *Applied Physics Reviews* 6.2 (2019), p. 021318.

[17] C. D. Bruzewicz, J. Chiaverini, R. McConnell, and J. M. Sage. "Trapped-ion quantum computing: Progress and challenges". *Applied Physics Reviews* 6.2 (2019), p. 021314.

[18] A. C. Y. Li et al. "Benchmarking variational quantum eigensolvers for the square-octagon-lattice Kitaev model". arXiv:2108.13375. 2021.

[19] G. J. Mooney, G. A. L. White, C. D. Hill, and L. C. L. Hollenberg. "Whole-Device Entanglement in a 65-Qubit Superconducting Quantum Computer". *Advanced Quantum Technologies* 4.10 (2021), p. 2100061.

[20] S. Endo, Z. Cai, S. C. Benjamin, and X. Yuan. "Hybrid Quantum-Classical Algorithms and Quantum Error Mitigation". *Journal of the Physical Society of Japan* 90.3 (2021), p. 032001.

[21] E. Farhi, J. Goldstone, and S. Gutmann. "A Quantum Approximate Optimization Algorithm". arXiv:1411.4028. 2014.

[22] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'Brien. "A variational eigenvalue solver on a photonic quantum processor". *Nature Communications* 5.4213 (2014).

[23] O. Higgott, D. Wang, and S. Brierley. "Variational Quantum Computation of Excited States". *Quantum* 3 (2019), p. 156.

[24] R. Santagati et al. "Witnessing eigenstates for quantum simulation of Hamiltonian spectra". *Science Advances* 4.1 (2018), eaap9646.

[25] K. M. Nakanishi, K. Mitarai, and K. Fujii. "Subspace-search variational quantum eigensolver for excited states". *Physical Review Research* 1.3 (2019), p. 033062.

[26] N. Yoshioka, Y. O. Nakagawa, K. Mitarai, and K. Fujii. "Variational quantum algorithm for nonequilibrium steady states". *Physical Review Research* 2.4 (2020), p. 043289.

[27] P. O'Malley et al. "Scalable Quantum Simulation of Molecular Energies". *Physical Review X* 6.3 (2016), p. 031007.

[28] J. R. McClean, J. Romero, R. Babbush, and A. Aspuru-Guzik. "The theory of variational hybrid quantum-classical algorithms". *New Journal of Physics* 18.2 (2016), p. 023023.

[29] J. R. McClean, S. Boixo, V. N. Smelyanskiy, R. Babbush, and H. Neven. "Barren plateaus in quantum neural network training landscapes". *Nature Communications* 9.4812 (2018).

[30] S. Wang, E. Fontana, M. Cerezo, K. Sharma, A. Sone, L. Cincio, and P. J. Coles. "Noise-induced barren plateaus in variational quantum algorithms". *Nature Communications* 12.6961 (2021).

[31] J. Kempe, A. Kitaev, and O. Regev. "The Complexity of the Local Hamiltonian Problem". *SIAM Journal on Computing* 35.5 (2006), pp. 1070–1097.

[32] B. O'Gorman, S. Irani, J. Whitfield, and B. Fefferman. "Electronic Structure in a Fixed Basis is QMA-complete". arXiv:2103.08215. 2021.

[33] L. Bittel and M. Kliesch. "Training Variational Quantum Algorithms is NP-Hard". *Physical Review Letters* 127.12 (2021), p. 120502.

[34] A. L. Blum and R. L. Rivest. "Training a 3-node neural network is NP-complete". *Neural Networks* 5.1 (1992), pp. 117–127.

[35] A. Kandala, A. Mezzacapo, K. Temme, M. Takita, M. Brink, J. M. Chow, and J. M. Gambetta. "Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets". *Nature* 549.7671 (2017), pp. 242–246.

[36] F. Arute et al. "Hartree-Fock on a superconducting qubit quantum computer". *Science* 369.6507 (2020), pp. 1084–1089.

[37] M. P. Harrigan et al. "Quantum approximate optimization of non-planar graph problems on a planar superconducting processor". *Nature Physics* 17 (2021), pp. 332–336.

[38] P. Huembeli and A. Dauphin. "Characterizing the loss landscape of variational quantum circuits". *Quantum Science and Technology* 6.2 (2021), p. 025011.

[39] M. C. Gutzwiller. "Effect of Correlation on the Ferromagnetism of Transition Metals". *Physical Review Letters* 10.5 (1963), pp. 159–162.

[40] J. Kanamori. "Electron Correlation and Ferromagnetism of Transition Metals". *Progress of Theoretical Physics* 30.3 (1963), pp. 275–289.

[41] D. P. Arovas, E. Berg, S. A. Kivelson, and S. Raghu. "The Hubbard Model". *Annual Review of Condensed Matter Physics* 13.1 (2021).

[42] S. Sachdev. "The Landscape of the Hubbard Model". *String Theory and Its Applications*. World Scientific, 2011.

[43] E. H. Lieb and F. Y. Wu. "Absence of Mott Transition in an Exact Solution of the Short-Range, One-Band Model in One Dimension". *Physical Review Letters* 20.25 (1968), pp. 1445–1448.

[44] H. Shiba. "Magnetic Susceptibility at Zero Temperature for the One-Dimensional Hubbard Model". *Physical Review B* 6.3 (1972), pp. 930–938.

[45] D. Wecker, M. B. Hastings, N. Wiebe, B. K. Clark, C. Nayak, and M. Troyer. "Solving strongly correlated electron models on a quantum computer". *Physical Review A* 92.6 (2015), p. 062318.

[46] J. LeBlanc et al. "Solutions of the Two-Dimensional Hubbard Model: Benchmarks and Results from a Wide Range of Numerical Algorithms". *Physical Review X* 5.4 (2015), p. 041041.

[47] D. Scalapino. "Numerical Studies of the 2D Hubbard Model". *Handbook of High-Temperature Superconductivity*. Springer, 2007, pp. 495–526.

[48] M. Qin, T. Schäfer, S. Andergassen, P. Corboz, and E. Gull. "The Hubbard Model: A Computational Perspective". *Annual Review of Condensed Matter Physics* 13.1 (2021).

[49] E. Dagotto. "Correlated electrons in high-temperature superconductors". *Reviews of Modern Physics* 66.3 (1994), pp. 763–840.

[50] S. Yamada, T. Imamura, and M. Machida. "16.447 TFlops and 159-Billion-dimensional Exact-diagonalization for Trapped Fermion-Hubbard Model on the Earth Simulator". *ACM/IEEE SC 2005 Conference*. IEEE, 2005.

[51] I. M. Georgescu, S. Ashhab, and F. Nori. "Quantum simulation". *Reviews of Modern Physics* 86.1 (2014), pp. 153–185.

[52] E. Altman et al. "Quantum Simulators: Architectures and Opportunities". *PRX Quantum* 2.1 (2021), p. 017003.

[53] T. Hensgens et al. "Quantum simulation of a Fermi-Hubbard model using a semiconductor quantum dot array". *Nature* 548.7665 (2017), pp. 70–73.

[54] L. Tarruell and L. Sanchez-Palencia. "Quantum simulation of the Hubbard model with ultracold fermions in optical lattices". *Comptes Rendus Physique* 19.6 (2018), pp. 365–393.

[55] C. Gross and I. Bloch. "Quantum simulations with ultracold atoms in optical lattices". *Science* 357.6355 (2017), pp. 995–1001.

[56] T. Jones, A. Brown, I. Bush, and S. C. Benjamin. "QuEST and High Performance Simulation of Quantum Computers". *Scientific Reports* 9.10736 (2019).

[57] C. Cade, L. Mineh, A. Montanaro, and S. Stanisic. "Strategies for solving the Fermi-Hubbard model on near-term quantum computers". *Physical Review B* 102.23 (2020), p. 235122.

[58] A. Montanaro, C. Cade, L. Mineh, and S. Stanisic. "Data from "Strategies for solving the Fermi-Hubbard model on near-term quantum computers"". `https://doi.org/10.5523/bris.1873owc1bcmrw1y4raeeuygzuy`. 2020.

[59] L. Mineh and A. Montanaro. "Solving the Hubbard model using density matrix embedding theory and the variational quantum eigensolver". *Physical Review B* 105.12 (2022), p. 125117.

[60] A. Montanaro and L. Mineh. "Data from "Solving the Hubbard model using density matrix embedding theory and the variational quantum eigensolver"". `https://doi.org/10.5523/bris.280vqefjhtg7k21ieyoyx9od3c`. 2021.

[61] L. Mineh and J. R. Scott. "C++ Quantum Simulation Library (QSL)". `https://github.com/lanamineh/qsl`. 2021.

[62] R. Nichols, L. Mineh, J. Rubio, J. C. F. Matthews, and P. A. Knott. “Designing quantum experiments with a genetic algorithm”. *Quantum Science and Technology* 4.4 (2019), p. 045012.

[63] Y. Nam et al. “Ground-state energy estimation of the water molecule on a trapped-ion quantum computer”. *npj Quantum Information* 6.33 (2020).

[64] C. Hempel et al. “Quantum Chemistry Calculations on a Trapped-Ion Quantum Simulator”. *Physical Review X* 8.3 (2018), p. 031022.

[65] R. Somma, G. Ortiz, J. E. Gubernatis, E. Knill, and R. Laflamme. “Simulating physical phenomena by quantum networks”. *Physical Review A* 65.4 (2002), p. 042323.

[66] I. D. Kivlichan, J. McClean, N. Wiebe, C. Gidney, A. Aspuru-Guzik, G. K.-L. Chan, and R. Babbush. “Quantum Simulation of Electronic Structure with Linear Depth and Connectivity”. *Physical Review Letters* 120.11 (2018), p. 110501.

[67] K. Setia, S. Bravyi, A. Mezzacapo, and J. D. Whitfield. “Superfast encodings for fermionic quantum simulation”. *Physical Review Research* 1.3 (2019), p. 033033.

[68] J. D. Whitfield, V. Havlíček, and M. Troyer. “Local spin operators for fermion simulations”. *Physical Review A* 94.3 (2016), p. 030301.

[69] C. Derby, J. Klassen, J. Bausch, and T. Cubitt. “Compact fermion to qubit mappings”. *Physical Review B* 104.3 (2021), p. 035118.

[70] M. Steudtner and S. Wehner. “Fermion-to-qubit mappings with varying resource requirements for quantum simulation”. *New Journal of Physics* 20.6 (2018), p. 063010.

[71] F Verstraete and J. I. Cirac. “Mapping local Hamiltonians of fermions to local Hamiltonians of spins”. *Journal of Statistical Mechanics: Theory and Experiment* 2005.09 (2005), P09012–P09012.

[72] R. C. Ball. “Fermions without Fermion Fields”. *Physical Review Letters* 95.17 (2005), p. 176407.

[73] S. B. Bravyi and A. Y. Kitaev. “Fermionic Quantum Computation”. *Annals of Physics* 298.1 (2002), pp. 210–226.

[74] K. Setia and J. D. Whitfield. "Bravyi-Kitaev Superfast simulation of electronic structure on a quantum computer". *The Journal of Chemical Physics* 148.16 (2018), p. 164104.

[75] A. Uvarov, J. D. Biamonte, and D. Yudin. "Variational quantum eigensolver for frustrated quantum systems". *Physical Review B* 102.7 (2020), p. 075104.

[76] M. Cerezo, A. Sone, T. Volkoff, L. Cincio, and P. J. Coles. "Cost function dependent barren plateaus in shallow parametrized quantum circuits". *Nature Communications* 12.1791 (2021).

[77] M. Ganzhorn et al. "Gate-Efficient Simulation of Molecular Eigenstates on a Quantum Computer". *Physical Review Applied* 11.4 (2019), p. 044092.

[78] S. A. Fischer and D. Gunlycke. "Symmetry Configuration Mapping for Compact Representation of Quantum Chemistry on Quantum Computers". arXiv: 1907.01493. 2019.

[79] R. J. Bartlett. "Coupled-cluster approach to molecular structure and spectra: a step toward predictive quantum chemistry". *The Journal of Physical Chemistry* 93.5 (1989), pp. 1697–1708.

[80] J. Romero, R. Babbush, J. R. McClean, C. Hempel, P. J. Love, and A. Aspuru-Guzik. "Strategies for quantum computing molecular energies using the unitary coupled cluster ansatz". *Quantum Science and Technology* 4.1 (2018), p. 014008.

[81] P. K. Barkoutsos et al. "Quantum algorithms for electronic structure calculations: Particle-hole Hamiltonian and optimized wave-function expansions". *Physical Review A* 98.2 (2018), p. 022322.

[82] A. J. McCaskey, Z. P. Parks, J. Jakowski, S. V. Moore, T. D. Morris, T. S. Humble, and R. C. Pooser. "Quantum chemistry as a benchmark for near-term quantum computers". *npj Quantum Information* 5.99 (2019).

[83] M. Urbanek, D. Camps, R. V. Beeumen, and W. A. de Jong. "Chemistry on Quantum Computers with Virtual Quantum Subspace Expansion". *Journal of Chemical Theory and Computation* 16.9 (2020), pp. 5425–5431.

[84] Y. Shen, X. Zhang, S. Zhang, J.-N. Zhang, M.-H. Yung, and K. Kim. "Quantum implementation of the unitary coupled cluster for simulating molecular electronic structure". *Physical Review A* 95.2 (2017), p. 020501.

[85] O. Shehab, K. Landsman, Y. Nam, D. Zhu, N. M. Linke, M. Keesan, R. C. Pooser, and C. Monroe. "Toward convergence of effective-field-theory simulations on digital quantum computers". *Physical Review A* 100.6 (2019), p. 062319.

[86] J. Lee, W. J. Huggins, M. Head-Gordon, and K. B. Whaley. "Generalized Unitary Coupled Cluster Wave functions for Quantum Computation". *Journal of Chemical Theory and Computation* 15.1 (2018), pp. 311–324.

[87] P.-L. Dallaire-Demers, J. Romero, L. Veis, S. Sim, and A. Aspuru-Guzik. "Low-depth circuit ansatz for preparing correlated fermionic states on a quantum computer". *Quantum Science and Technology* 4.4 (2019), p. 045005.

[88] I. G. Ryabinkin, T.-C. Yen, S. N. Genin, and A. F. Izmaylov. "Qubit Coupled Cluster Method: A Systematic Approach to Quantum Chemistry on a Quantum Computer". *Journal of Chemical Theory and Computation* 14.12 (2018), pp. 6317–6326.

[89] B. T. Gard, L. Zhu, G. S. Barron, N. J. Mayhall, S. E. Economou, and E. Barnes. "Efficient symmetry-preserving state preparation circuits for the variational quantum eigensolver algorithm". *npj Quantum Information* 6.10 (2020).

[90] H. R. Grimsley, S. E. Economou, E. Barnes, and N. J. Mayhall. "An adaptive variational algorithm for exact molecular simulations on a quantum computer". *Nature Communications* 10.3007 (2019).

[91] H. L. Tang, V. Shkolnikov, G. S. Barron, H. R. Grimsley, N. J. Mayhall, E. Barnes, and S. E. Economou. "Qubit-ADAPT-VQE: An Adaptive Algorithm for Constructing Hardware-Efficient Ansätze on a Quantum Processor". *PRX Quantum* 2.2 (2021), p. 020310.

[92] D. Wecker, M. B. Hastings, and M. Troyer. "Progress towards practical quantum variational algorithms". *Physical Review A* 92.4 (2015), p. 042303.

[93] J.-M. Reiner, F. Wilhelm-Mauch, G. Schön, and M. Marthaler. "Finding the ground state of the Hubbard model by variational methods on a quantum computer with gate errors". *Quantum Science and Technology* 4.3 (2019), p. 035005.

[94] Z. Cai. "Resource Estimation for Quantum Variational Simulations of the Hubbard Model". *Physical Review Applied* 14.1 (2020), p. 014059.

[95] N. Vogt, S. Zanker, J.-M. Reiner, T. Eckl, A. Marusczyk, and M. Marthaler. "Preparing symmetry broken ground states with variational quantum algorithms". arXiv:2007.01582. 2020.

[96] R. Wiersema, C. Zhou, Y. de Sereville, J. F. Carrasquilla, Y. B. Kim, and H. Yuen. "Exploring Entanglement and Optimization within the Hamiltonian Variational Ansatz". *PRX Quantum* 1.2 (2020), p. 020319.

[97] J. L. Bosse and A. Montanaro. "Probing ground state properties of the kagome antiferromagnetic Heisenberg model using the Variational Quantum Eigensolver". arXiv:2108.08086. 2021.

[98] J. Kattemölle and J. van Wezel. "Variational quantum eigensolver for the Heisenberg antiferromagnet on the kagome lattice". arXiv:2108.02175. 2021.

[99] M. Larocca, P. Czarnik, K. Sharma, G. Muraleedharan, P. J. Coles, and M. Cerezo. "Diagnosing barren plateaus with tools from quantum optimal control". arXiv:2105.14377. 2021.

[100] Z. Jiang, K. J. Sung, K. Kechedzhi, V. N. Smelyanskiy, and S. Boixo. "Quantum Algorithms to Simulate Many-Body Physics of Correlated Fermions". *Physical Review Applied* 9.4 (2018), p. 044036.

[101] W. J. Huggins, J. R. McClean, N. C. Rubin, Z. Jiang, N. Wiebe, K. B. Whaley, and R. Babbush. "Efficient and noise resilient measurements for quantum chemistry on near-term quantum computers". *npj Quantum Information* 7.23 (2021).

[102] O. Crawford, B. van Straaten, D. Wang, T. Parks, E. Campbell, and S. Brierley. "Efficient quantum measurement of Pauli operators in the presence of finite sampling error". *Quantum* 5 (2021), p. 385.

[103] P. Gokhale, O. Angiuli, Y. Ding, K. Gui, T. Tomesh, M. Suchara, M. Martonosi, and F. T. Chong. "$O(N^3)$ Measurement Cost for Variational Quantum Eigensolver on Molecular Hamiltonians". *IEEE Transactions on Quantum Engineering* 1 (2020), pp. 1–24.

[104] A. F. Izmaylov, T.-C. Yen, and I. G. Ryabinkin. "Revising the measurement process in the variational quantum eigensolver: is it possible to reduce the number of separately measured operators?" *Chemical Science* 10.13 (2019), pp. 3746–3755.

[105] P. Gokhale, O. Angiuli, Y. Ding, K. Gui, T. Tomesh, M. Suchara, M. Martonosi, and F. T. Chong. "Minimizing State Preparations in Variational Quantum Eigensolver by Partitioning into Commuting Families". arXiv:1907.13623. 2019.

[106] I. Hamamura and T. Imamichi. "Efficient evaluation of quantum observables using entangled measurements". *npj Quantum Information* 6.56 (2020).

[107] R. Barends et al. "Superconducting quantum circuits at the surface code threshold for fault tolerance". *Nature* 508.7497 (2014), pp. 500–503.

[108] T. Walter et al. "Rapid High-Fidelity Single-Shot Dispersive Readout of Superconducting Qubits". *Physical Review Applied* 7.5 (2017), p. 054020.

[109] W. Lavrijsen, A. Tudor, J. Müller, C. Iancu, and W. de Jong. "Classical Optimizers for Noisy Intermediate-Scale Quantum Devices". *2020 IEEE International Conference on Quantum Computing and Engineering (QCE)*. 2020, pp. 267–277.

[110] K. J. Sung, J. Yao, M. P. Harrigan, N. C. Rubin, Z. Jiang, L. Lin, R. Babbush, and J. R. McClean. "Using models to improve optimizers for variational quantum algorithms". *Quantum Science and Technology* 5.4 (2020), p. 044008.

[111] G. Verdon, M. Broughton, J. R. McClean, K. J. Sung, R. Babbush, Z. Jiang, H. Neven, and M. Mohseni. "Learning to learn with quantum neural networks via classical neural networks". arXiv:1907.05415. 2019.

[112] M. Wilson, R. Stromswold, F. Wudarski, S. Hadfield, N. M. Tubman, and E. G. Rieffel. "Optimizing quantum heuristics with meta-learning". *Quantum Machine Intelligence* 3.13 (2021).

[113] J. M. Kübler, A. Arrasmith, L. Cincio, and P. J. Coles. "An Adaptive Optimizer for Measurement-Frugal Variational Algorithms". *Quantum* 4 (2020), p. 263.

[114] A. Arrasmith, L. Cincio, R. D. Somma, and P. J. Coles. "Operator Sampling for Shot-frugal Optimization in Variational Algorithms". arXiv:2004.06252. 2020.

[115] J. C. Spall. "Introduction to Stochastic Search and Optimization". John Wiley & Sons, Inc., 2003.

[116] J. C. Spall. "An overview of the simultaneous perturbation method for efficient optimization". *Johns Hopkins APL Techincal Digest* 19.4 (1998).

[117] A. Montanaro and S. Stanisic. “Compressed variational quantum eigensolver for the Fermi-Hubbard model”. arXiv:2006.01179. 2020.

[118] J. Spall. “Implementation of the simultaneous perturbation algorithm for stochastic optimization”. *IEEE Transactions on Aerospace and Electronic Systems* 34.3 (1998), pp. 817–823.

[119] K. M. Nakanishi, K. Fujii, and S. Todo. “Sequential minimal optimization for quantum-classical hybrid algorithms”. *Physical Review Research* 2.4 (2020), p. 043158.

[120] M. Ostaszewski, E. Grant, and M. Benedetti. “Structure optimization for parameterized quantum circuits”. *Quantum* 5 (2021), p. 391.

[121] R. M. Parrish, J. T. Iosue, A. Ozaeta, and P. L. McMahon. “A Jacobi Diagonalization and Anderson Acceleration Algorithm For Variational Quantum Algorithm Parameter Optimization”. arXiv:1904.03206. 2019.

[122] P. Tseng. “Convergence of a Block Coordinate Descent Method for Nondifferentiable Minimization”. *Journal of Optimization Theory and Applications* 109.3 (2001), pp. 475–494.

[123] P. Suchsland, P. K. Barkoutsos, I. Tavernelli, M. H. Fischer, and T. Neupert. “Simulating a ring-like Hubbard system with a quantum computer”. arXiv:2104.06428. 2021.

[124] F. Verstraete, J. I. Cirac, and J. I. Latorre. “Quantum circuits for strongly correlated quantum systems”. *Physical Review A* 79.3 (2009), p. 032316.

[125] M. Vekić and S. R. White. “Hubbard model with smooth boundary conditions”. *Physical Review B* 53.21 (1996), pp. 14552–14557.

[126] S. G. Johnson. “The NLopt nonlinear-optimization package”. `http://github.com/stevengj/nlopt`.

[127] H. Ma, M. Govoni, and G. Galli. “Quantum simulations of materials on near-term quantum computers”. *npj Computational Materials* 6.85 (2020).

[128] I. Rungger et al. “Dynamical mean field theory algorithm and experiment on quantum computers”. arXiv:1910.04735. 2019.

[129] N. Sheng, C. Vorwerk, M. Govoni, and G. Galli. “Quantum Embedding Theories to Simulate Condensed Systems on Quantum Computers”. arXiv:2105.04736. 2021.

[130] B. Bauer, D. Wecker, A. J. Millis, M. B. Hastings, and M. Troyer. "Hybrid Quantum-Classical Approach to Correlated Materials". *Physical Review X* 6.3 (2016), p. 031045.

[131] F. Barratt, J. Dborin, M. Bal, V. Stojevic, F. Pollmann, and A. G. Green. "Parallel quantum simulation of large systems on small NISQ computers". *npj Quantum Information* 7.79 (2021).

[132] S.-H. Lin, R. Dilip, A. G. Green, A. Smith, and F. Pollmann. "Real- and Imaginary-Time Evolution with Compressed Quantum Circuits". *PRX Quantum* 2.1 (2021), p. 010342.

[133] M. Foss-Feig, D. Hayes, J. M. Dreiling, C. Figgatt, J. P. Gaebler, S. A. Moses, J. M. Pino, and A. C. Potter. "Holographic quantum algorithms for simulating correlated spin systems". *Physical Review Research* 3.3 (2021), p. 033002.

[134] G. Knizia and G. K.-L. Chan. "Density Matrix Embedding: A Simple Alternative to Dynamical Mean-Field Theory". *Physical Review Letters* 109.18 (2012), p. 186404.

[135] G. Knizia and G. K.-L. Chan. "Density Matrix Embedding: A Strong-Coupling Quantum Embedding Theory". *Journal of Chemical Theory and Computation* 9.3 (2013), pp. 1428–1432.

[136] B.-X. Zheng and G. K.-L. Chan. "Ground-state phase diagram of the square lattice Hubbard model from density matrix embedding theory". *Physical Review B* 93.3 (2016), p. 035126.

[137] I. W. Bulik, G. E. Scuseria, and J. Dukelsky. "Density matrix embedding from broken symmetry lattice mean fields". *Physical Review B* 89.3 (2014), p. 035140.

[138] N. C. Rubin. "A Hybrid Classical/Quantum Approach for Large-Scale Studies of Quantum Systems with Density Matrix Embedding Theory". arXiv: 1610.06910. 2016.

[139] T. Yamazaki, S. Matsuura, A. Narimani, A. Saidmuradov, and A. Zaribafiyan. "Towards the Practical Application of Near-Term Quantum Computers in Quantum Chemistry Simulations: A Problem Decomposition Approach". arXiv:1806.01305. 2018.

[140] Y. Kawashima et al. "Optimizing electronic structure simulations on a trapped-ion quantum computer using problem decomposition". *Communications Physics* 4.245 (2021).

[141] J. Tilly et al. "Reduced density matrix sampling: Self-consistent embedding and multiscale electronic structure on current generation quantum computers". *Physical Review Research* 3.3 (2021), p. 033230.

[142] S. Wouters, C. A. Jiménez-Hoyos, Q. Sun, and G. K.-L. Chan. "A Practical Guide to Density Matrix Embedding Theory in Quantum Chemistry". *Journal of Chemical Theory and Computation* 12.6 (2016), pp. 2706–2719.

[143] I. W. Bulik, W. Chen, and G. E. Scuseria. "Electron correlation in solids via density embedding theory". *The Journal of Chemical Physics* 141.5 (2014), p. 054113.

[144] B. Bamieh. "Discovering Transforms: A Tutorial on Circulant Matrices, Circular Convolution, and the Discrete Fourier Transform". arXiv:1805.05533. 2018.

[145] P. Delsarte and Y. Genin. "Spectral properties of finite Toeplitz matrices". *Mathematical Theory of Networks and Systems*. Springer-Verlag, 1984, pp. 194–213.

[146] H. J. Landau. "The inverse eigenvalue problem for real symmetric Toeplitz matrices". *Journal of the American Mathematical Society* 7.3 (1994), pp. 749–749.

[147] A. Cantoni and P. Butler. "Properties of the Eigenvectors of Persymmetric Matrices with Applications to Communication Theory". *IEEE Transactions on Communications* 24.8 (1976), pp. 804–809.

[148] D. S. Steiger, T. Häner, and M. Troyer. "ProjectQ: an open source software framework for quantum computing". *Quantum* 2 (2018), p. 49.

[149] M. S. Anis et al. "Qiskit: An Open-source Framework for Quantum Computing". 2021.

[150] Cirq Developers. "Cirq". 2021.

[151] M. Fingerhuth, T. Babej, and P. Wittek. "Open source software in quantum computing". *PLoS One* 13.12 (2018), e0208561.

[152] T. Häner and D. S. Steiger. "0.5 petabyte simulation of a 45-qubit quantum circuit". *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. ACM, 2017.

[153] B. Villalonga, S. Boixo, B. Nelson, C. Henze, E. Rieffel, R. Biswas, and S. Mandrà. "A flexible high-performance simulator for verifying and benchmarking quantum circuits implemented on real hardware". *npj Quantum Information* 5.86 (2019).

[154] A. Fatima and I. L. Markov. "Faster Schrödinger-style simulation of quantum circuits" (2020). arXiv:2008.00216.

[155] Quantum AI team and collaborators. "qsim". 2020.

[156] Y. Suzuki et al. "Qulacs: a fast and versatile quantum circuit simulator for research purpose". *Quantum* 5 (2021), p. 559.

[157] G. G. Guerreschi, J. Hogaboam, F. Baruffa, and N. P. D. Sawaya. "Intel Quantum Simulator: a cloud-ready high-performance simulator of quantum circuits". *Quantum Science and Technology* 5.3 (2020), p. 034007.

[158] M. Smelyanskiy, N. P. D. Sawaya, and A. Aspuru-Guzik. "qHiPSTER: The Quantum High Performance Software Testing Environment". arXiv:1601.07195. 2016.

[159] V. Gheorghiu. "Quantum++: A modern C++ quantum computing library". *PLoS One* 13.12 (2018), e0208073.

[160] D. A. Patterson and J. L. Hennessy. "Computer Organization and Design RISC-V Edition". Cambridge, MA: Morgan Kaufmann Publishers, 2018.

[161] A. Fog. "Instruction tables: Lists of instruction latencies, throughputs and micro-operation breakdowns for Intel, AMD and VIA CPUs". *Copenhagen University College of Engineering* 93 (2011), p. 110.

[162] D. Knuth. "The Art of Computer Programming, Volume 3: Sorting and Searching". Pearson Education, 1998.

[163] D. Knuth. "The Art of Computer Programming, Volume 2: Seminumerical Algorithms". Pearson Education, 2014.

[164] N. C. Rubin, K. Gunst, A. White, L. Freitag, K. Throssell, G. K.-L. Chan, R. Babbush, and T. Shiozaki. "The Fermionic Quantum Emulator". *Quantum* 5 (2021), p. 568.

[165] S. Jaques and T. Häner. "Leveraging state sparsity for more efficient quantum simulations". arXiv:2105.01533. 2021.

[166] S. E. Anderson. *Bit Twiddling Hacks*. 2005. URL: `https://graphics.stanford.edu/~seander/bithacks.html#NextBitPermutation` (visited on 04/10/2021).