Tensor network inspired methods for near-term quantum computation

VINUL WIMALAWEERA

UNIVERSITY COLLEGE LONDON

DEPARTMENT OF PHYSICS & ASTRONOMY

Submitted to University College London (UCL) in partial fulfilment of the requirements for the award of the degree of Doctor of Philosophy.

Primary supervisor:	Prof. Andrew G. Green
Secondary supervisor:	Prof. Dan Browne
Examining committee:	Prof. Andrew Daley
	Prof. Arijeet Pal

Declaration

I, **Vinul Wimalaweera**, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the thesis.

Vinul Wimalaweera
London, United Kingdom
02/09/2024

Abstract

This thesis investigates how tensor networks can inform our design of algorithms for near-term quantum devices. We primarily work in the paradigm of variational quantum algorithms where tensor networks can be used as a variational model ansatz tuned to the underlying problem's entanglement structure. This allows for the efficient use of resources, a problem of particular importance for today's small-scale noisy quantum computers. In particular, this thesis focuses on studying quantum tensor networks for quantum simulation and machine learning. We develop algorithms designed to simulate large quantum systems in the thermodynamic limit, implementing tools from classical tensor network literature which allow us to represent these infinite states using finite circuits. In doing so, we design an algorithm to study the groundstate properties of the transverse field Ising model through the quantum critical point. We also design and implement a time evolution algorithm that utilises this infinite tensor network ansatz. The design of the algorithm allows for portability between various architectures, a feature we demonstrate by testing the algorithm on both Google's superconducting and Quantinuum's trapped-ion architectures. Collaborative benchmarking in this way provides data that can be used to investigate impacts of device specific characteristics such as device error or shot budgets on the performance of the algorithm. In addition, the testing highlighted specific areas of improvements for current generation device which provides vital information for hardware teams on directions towards quantum advantage. Tensor network ansatzes also provide a model family of interest for quantum machine learning. We utilise the close correspondence between classical and quantum tensor networks to demonstrate a heuristic technique to mitigate barren plateaus, a problem of significant importance for any quantum machine learning model. Subsequently, we propose a native tensor network algorithm to find supervised learning models. In summary, this thesis leverages the utility of the rich literature developed in classical tensor networks to design near-term quantum algorithms in a range of fields.

Impact Statement

Current generation quantum computers are small and noisy, which limits their usefulness in applications such as quantum simulation and machine learning. We develop techniques which leverage classical tensor network ideas to develop quantum algorithms tailored to near-term devices. Our focus is on solving quantum simulation problems, including ground state optimisation and time evolving a translationally invariant tensor network ansatz. Additionally, we utilise this tensor network approach to address quantum machine learning problems, including initialising quantum models and devising novel techniques for generating classifiers.

Within academia, the impact of this work is a better understanding of the connection between tensor networks and quantum circuits, potentially as an avenue to achieve quantum utility. We have undertaken experiments using state-of-the-art quantum devices owned by Google and Quantinuum to showcase and benchmark their quality. In doing so we provide end user insight into the utility of their devices for questions of interest in condensed matter physics, with methods that can be directly compared to classical techniques. Ultimately, the goal of this line of research is to develop protocols for using quantum computers to address vital challenges such as materials discovery. The work included in this thesis, though removed from this goal, aims to provide some preliminary insights into studying large quantum systems on a quantum device.

Acknowledgements

I am extremely grateful to everyone that contributed to the completion of this thesis. Firstly, I would like to thank my supervisor, Professor Andrew Green, for his mentorship throughout the PhD. His valuable insights, careful guidance and unwavering support are the cornerstone of this thesis.

I am also thankful to the members of the Green group, past and present, for their discussions and camaraderie. In particular, I would like to extend my heartfelt thanks to Lesley, Jamie, Fergus, Lewis, and Brian, whose collaborations were invaluable to the development of this thesis.

Completing this thesis was not just an academic challenge. I want to thank all my friends. They provided me with distractions when I needed it and support when I sought it. These past four years would have been much more daunting without you. I would like to give special thanks to Kasia; our writing club made the thesis writing process infinitely more bearable.

Finally, I owe a debt of gratitude to my family, whose love and support have been a constant source of strength. To my brother, whose companionship provided a continuous source of comfort. To my girlfriend, Kav, whose unwavering tenderness and love kept me going. And to my parents, my mum, who always takes care of me and my dad, who shaped me into the person I am today. I hope I have made you proud.

I dedicate this thesis to my beloved mum and dad.

Contents

Table of Acronyms

Notation	Description
VQA	variational quantum algorithm
DQPT	dynamical quantum phase transition
TFIM	transverse field Ising model
iMPS	translationally invariant matrix product state
TEBD	time-evolving block decimation
TDVP	time dependent variational principle
MPS	matrix product state
PEPS	projected entangled pair state
MERA	multiscale entanglement renormalisation ansatz
MPO	matrix product operator
SVD	singular value decomposition
SPSA	simultaneous perturbation stochastic approximation
QML	quantum machine learning
PQC	parameterised quantum circuits
DMRG	density matrix renormalization group
PCA	principal component analysis
SVM	support vector machine
NISQ	noisy intermediate-scale quantum
DMT	density matrix truncation
DAOE	dissipation-assisted operator evolution
TTN	tree tensor network

List of Figures

List of Tables

Chapter 1

Introduction

Computing technology is a central tool for our ability to understand the physical world. It allows us to make predictions about systems at scale with incredible accuracy. This predictive capability and understanding is the driver behind many modern technological advancements. Our progress is built not just upon the development of novel algorithms, but also on the design and manufacture of better chips. Modern computers use binary logic gates and bits to perform computation. We have steadily reduced the size of these components and improved their processing capabilities to produce orders of magnitude improvements in their processing capabilities since their original implementation. However, in the case of simulating the physical world, the game is still rigged against us.

Fundamentally, nature does not compute using bits. Our current best model for the physical world, particularly when it comes to simulating matter, comes from quantum mechanics. For systems of modest size, brute force computation requires resources beyond the capabilities of even our largest supercomputers. In fact, simulating quantum systems directly is a fundamental limitation of modern computers, in general requiring resources which scale exponentially in the system size. To circumvent this issue we consider shifting our paradigm of computing. By taking inspiration from nature, perhaps we can build a new kind of computer.

A quantum computer is a computer whose fundamental unit of information is a quantum system, often referred to as a qubit. Its operation is modelled under the laws of quantum mechanics. This is not just a replacement for classical computing, this is a different model of computation altogether. In a quantum computer, the majority of the information is stored in the correlations between parts of the quantum system, quantified as the system's entanglement. Classical comput-

ers struggle to simulate systems with significant entanglement as the classical description of systems with significant entanglement is prohibitively large, thus limiting its ability to simulate interesting quantum matter. However, quantum computers have the potential to utilise the laws of quantum mechanics to generate large amounts of entanglement, thereby providing a route to solve problems that may require entanglement such as simulating quantum matter. This is not just a theoretical proposition; our ability to control quantum systems has developed to the degree that we can build small-scale, noisy quantum computers.

Building a quantum computer is hard. To maintain entanglement, particles the size of individual atoms have to be completely isolated from their surroundings. Miniscule temperature fluctuations and less than perfect isolation of these qubits can result in the loss of any useable information through decoherence. However, we also have to be able to control the particles, breaking this isolation briefly to apply the quantum gates necessary for us to perform the computation. These fundamentally contradictory requirements mean that building a quantum computer is a seemingly impossible task. However, in recent years, we have been able to do just that. There are currently many platforms that have 10s-100s of noisy qubits. The question now is, can we do anything useful with these devices? Quantum computing technology has promised many novel theoretical feats, from simulating materials to breaking cryptographic protocols. The race is now on to fulfil some of these promises.

1.1 Outline of the thesis

This thesis proposes approaches to simulating condensed matter systems using near-term quantum devices. The size and performance of these devices require efficient use of resources. To do so, we use insights from a branch of classical computational methods known as tensor networks to design these quantum algorithms. We demonstrate that these insights are not just applicable in the context of quantum simulation, but also for the purposes of quantum machine learning.

- **Chapter 1** introduces the background necessary for the remainder of the thesis. In particular, we focus on the basics of quantum computation and tensor networks.

The remainder of this thesis is divided into two parts. The first part discusses utilising quantum circuits representing tensor networks, which we refer to as quantum tensor networks, for quantum simulation.

- **Chapter 2** presents quantum tensor networks as a variational ansatz for groundstate optimisation in the thermodynamic limit.
- **Chapter 3** utilises the previous quantum tensor network ansatz to perform time evolution. We test this algorithm on a quantum device by attempting to reproduce the dynamical quantum phase transition of the transverse field Ising model.
- **Chapter 4** develops on the previous chapter by proposing a novel classical and quantum time evolution algorithm that preserves local information during the evolution.

The second part of this thesis applies insights from tensor networks to problems in quantum machine learning.

- **Chapter 5** considers the problem of initialising a quantum machine learning model appropriately to avoid common pitfalls in training these models.
- **Chapter 6** proposes an algorithm for deterministically generating a tensor network classifier and further refining it.

1.2 Quantum computation

1.2.1 Introduction to quantum computing

Computing, in its broadest form, involves designing and implementing physical devices that process information; it is the bedrock of developing technologies. Modern computers used in practice today, which we call classical computers, utilise macroscopic properties of materials to implement binary logic gates on vast scales. This computational model has had great success, including aiding in our understanding of quantum physics. However, the modern processor and binary logic gates are not the only computers available to us. Since the 1980s, it has been proposed that quantum systems could form a fundamental unit of computation, with a model of computation utilising the laws of quantum mechanics.

This field, called quantum computation or quantum information, has been subsequently studied in great theoretical detail. Initial results included the development of universal quantum computation [1, 2] showing that this model of computation could produce a universal computer. A primary focus of quantum computing from its inception was the desire to simulate quantum systems directly. Classically, this problem is challenging as the resources required to simulate quantum systems naively scale exponentially with system size. As quantum computers rely on quantum mechanics to process information, they are proposed as offering a way to simulate large quantum systems directly. In addition, it is proposed that quantum computers may have much broader impacts, with a zoo of algorithms proposed in many fields, including machine learning, optimisation and cryptography [3]. A full review of quantum computing is beyond the scope of this thesis, but in the following section, I will introduce the fundamental building blocks necessary for the remainder of this thesis.

Fundamentals of quantum computing

Though several models of quantum computation exist, we focus on the digital quantum circuit model with qubits. The basic building blocks of this paradigm are qubits, quantum gates, and measurements. This model also includes a diagrammatic language, which we outline in parallel in the discussion below. From a mathematical perspective, a qubit is a 2-dimensional complex vector (a, b) where the elements of the vector have a norm of 1 i.e. $|a|^2+|b|^2= 1$. In the Dirac bra-ket notation, they

are often represented as acting on the $|0\rangle$ and $|1\rangle$ basis states. Therefore, a general qubit can be written as $|\psi\rangle = a\,|0\rangle + b\,|1\rangle$. As multiple qubits are combined, the tensor outer product is taken to represent the system's overall state. This means that the space required to represent an arbitrary state of n qubits scales as 2^n. The qubits are normally initialised to the $|0\rangle$ state in digital quantum computers. A set of $n = 3$ qubits initialised as such are represented as

$$|0\rangle^{\otimes 3} = \begin{array}{l} |0\rangle \text{ ——} \\ |0\rangle \text{ ——} \\ |0\rangle \text{ ——} \end{array} . \tag{1.2.1}$$

As quantum computers obey the laws of quantum mechanics, unitary operations represent general processes which manipulates information on a quantum computer. A unitary U acting on n qubits can be represented by a complex matrix of size $2^n \times 2^n$ and satisfies the property $UU^\dagger = U^\dagger U = I$. Depending on the particular universal gate set of a quantum device, the native gates available to a user of a quantum computer vary. Generally gates are represented diagrammatically as rectangles with annotation. There are many common gates which come up throughout this thesis. Firstly, the Pauli rotation matrices, which are exponentiations of the Pauli matrices X, Y and Z, are given by

$$R_x(\theta) \equiv e^{-i\theta x/2} = \text{——}\boxed{R_x}\text{——} , \tag{1.2.2}$$

where $x \in \{X, Y, Z\}$ and θ is a rotation angle. A Hadmard gate is also a single qubit gate given by

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} = \text{——}\boxed{H}\text{——} . \tag{1.2.3}$$

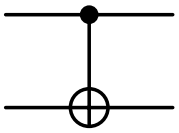
Finally, common 2-qubit gates are the controlled operation gates. This is where the action on the second (target) qubit is only applied if the first (control) qubit is in the state $|1\rangle$. A controlled operation which applies an arbitrary single-qubit unitary U is represented by

$$CU = \begin{array}{c} \bullet \\ \boxed{U} \end{array} . \tag{1.2.4}$$

A common example of this type of operation is the $CNOT$ gate, which flips the state of the target qubit if the control qubit is $|1\rangle$. The $CNOT$

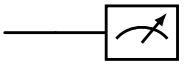
is often represented as

$$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} = \quad . \tag{1.2.5}$$



Once a quantum computer has processed the information, it has to be extracted. The process of extraction is done using quantum measurements. In general, all measurements in this thesis are projective measurements. Projective measurements measure a qubit in the basis of a Hermitian operator. In most instances, this will be specified by the computational basis of the quantum device. A projective measurement is represented diagrammatically as

$$meas = \quad . \tag{1.2.6}$$



Not all the qubits used in the computation will be measured in some instances. In these cases, the computation can be represented in the density matrix formalism, with a partial trace being applied over the qubits that have not been measured.

Many algorithms have been derived to solve various tasks using this digital quantum circuit formalism. However, implementing these algorithms is not simply a theoretical question. Quantum computing, in practice, relies on the careful control of quantum systems and their interactions. The current iteration of devices has imperfect control, resulting in large amounts of noise. The following section introduces these types of devices, which form the focus of this thesis.

1.2.2 Near-term quantum computing

Generally, quantum algorithms with theoretical guarantees of solving problems beyond the scope of current classical computers require fault-tolerant quantum computers. Full fault-tolerant quantum computing requires overcoming noise when performing fine control over quantum systems. Doing so not only requires refining experimental processes, due to the sensitivity of qubits to their environment and the likelihood of decoherence [4] it is likely that processing quantum information in a fault-tolerant manner involves the use of quantum error correction protocols; where a number of physical qubits are used redundantly to produce a logical qubit [5]. However, quantum error correction at scale

is beyond the capabilities of current-generation quantum devices. This is partly because a single error-corrected logical qubit requires many physical qubits. Scaling qubit numbers whilst maintaining or improving qubit quality is exceptionally challenging. In recent years, there have been proof of concept experiments for several error-corrected qubits on various platforms. However, scaling this technology up is a non-trivial problem [6].

In the meantime, the utility of existing quantum devices is an open research question. Across various architectures and research groups, there are currently many quantum devices that contain of order $10^2 - 10^3$ physical qubits. Examples include IBM's superconducting architecture with 127 physical qubits [7], Quantinuum's trapped-ion architecture with 56 physical qubits [8] and QuEra's neutral atom architecture with 280 physical qubits [9, 10]. The term often used to refer to these devices are noisy intermediate-scale quantum (NISQ) devices [11]. As we are working with physical qubits on these devices, the noise and behaviour of these qubits vary significantly depending on the platform. Designing noisy intermediate-scale quantum (NISQ) algorithms requires careful consideration of this noise and an adaptable algorithm that can accommodate devices with different characteristics.

A broad range of underlying technologies is utilised to design and build NISQ devices that generally determine their performance [12]. For the purposes of this thesis I focus on trapped-ion devices and superconducting devices as these were the architectures available to us for our experiments. The original design of trapped-ion quantum computers were ions in a RF Paul trap where the energy levels of the ions served as the qubit states whilst the shared ion motional modes were used as a quantum bus between qubits [13, 14]. The benefit of these devices is that the qubits generally have relatively long coherence times, with some ions having hyperfine coherence times of around 50s, with limited depolarising errors as long as the number of qubits are well controlled. Additionally, since all the qubits are fundamentally identical, they are not affected by fabrication errors in the ways that superconducting circuits are. In addition, platforms like that of Quantinuum's H-series have advantageous protocols such as all-to-all connectivity and mid-circuit measurement [8]. However, scaling this technology to a high number of qubits is generally quite challenging, with control of increasing numbers of qubits becoming less reliable. In addition, it is known that scaling the number of ions in a chain leads to the crowding of the oscillation spectrum. Beyond a point it

becomes challenging to single out one motional mode which impacts performance and can lead to strong decoherence [15]. Finally, readout times can be slow with trapped-ion technologies, with readouts times being of the order of μs [16]. In contrast, superconducting devices like that of Google's Sycamore processors are generally relatively scalable, whereby they can leverage pre-existing semiconductor fabrication technologies. The underlying design of superconducting qubits can vary significantly but there are three main qubit architectures, namely charge qubits, flux qubits and phase qubits [17]. One of the most popular design for qubits is the Transmon qubit comprised of two Josephson junctions shunted by capacitors [18]. Generally, superconducting quantum computers are highly configurable and they are readily fabricated and controlled using existing semiconductor and microwave control technology [17]. However, unlike trapped-ion technology qubit connectivity of superconducting devices is fixed by the topology of the device. They are often larger than trapped-ion quantum computer and require significant cooling to operate [15]. In addition, superconducting qubits generally have a short coherence time of order ms [19]. Decoherence effects and crosstalk are exacerbated as the qubits scale and must be carefully considered when designing superconducting devices [20]. Beyond trapped-ion and superconducting devices, there remains a broad array of experimental approaches to designing and building quantum computers, including technology based on neutral atoms [10] and photonics [21].

With the rise in near-term quantum devices came the experimental push to design experiments that show a quantum advantage. Though diverse in their attempts, quantum advantage experiments look to demonstrate a computation that can readily be performed on current-generation quantum devices that would be intractable to compute classically. One of the first claims of quantum advantage came from the Google AI Quantum team in 2019, where they used their 53 qubit Sycamore processor to perform random circuit sampling [22]. Following this, in 2020, the Jiuzhang photonic quantum computing platform performed Gaussian boson sampling [21]. Both these experiments focus on problems designed around the specific device's characteristics. Quantum advantage in general refers to a provable algorithmic speed-up to known classical methods. There exists a more experimentally focussed notion, referred to as quantum utility, which looks to use quantum computers for solve specific physical or useful problems better than all known classical algorithms. This standard means that a quantum al-

gorithm which demonstrates quantum utility could be used in place of the analogous classical algorithm to solve a problem of practical usefulness. A recent example of an experiment where quantum utility is the explicit goal is the work by the IBM Quantum group in 2023 [7], where they looked to calculate expectation values and perform Trotterised time evolution of a 2D transverse-field Ising model. Despite the variety of approaches and increasing frequency of results, these experiments have not unanimously demonstrated quantum advantage or utility. Following these experimental results, classical algorithms have been proposed theoretically and experimentally, which either match or outperform the outcomes of these results. In particular, tensor networks have been a potent tool when performing numerical simulations of these experiments [23, 24].

Many NISQ algorithms rely on a hybrid quantum-classical paradigm often referred to as variational quantum algorithms. These algorithms rely on encoding the problem they are trying to solve as an optimisation problem where a parameterised quantum circuit acts as part of the model to be optimised. The following section contains an outline of variational quantum algorithms.

1.2.3 Variational quantum algorithms

Variational quantum algorithms (VQAs) refer to a broad range of algorithms which utilise a hybrid quantum-classical approach to solve problems. They have been particularly popular on NISQ devices due to their ability to overcome some of the shortcomings of NISQ devices, such as their size and noisiness. In addition, variational quantum algorithms (VQAs) have a general purpose structure, making them easy to adapt to several different problems. Algorithms that bear the structure of VQAs have been proposed in fields ranging from groundstate and dynamical simulation to error correction and compilation [25].

VQAs utilise parameterised quantum circuits to encode the target problem as the optima of an objective function. The optimisation itself is done classically and utilises a number of tools readily available in classical computing, particularly in well-developed fields like machine learning. The parameterised quantum circuits within the objective function essentially act as a model class in a similar manner to a neural network within machine learning. This section outlines the basic building blocks of VQAs, including constructing an objective function, the parameterised quantum circuit ansatz and their optimisation. Figure 1.1 shows an outline of these steps. In addition, despite the

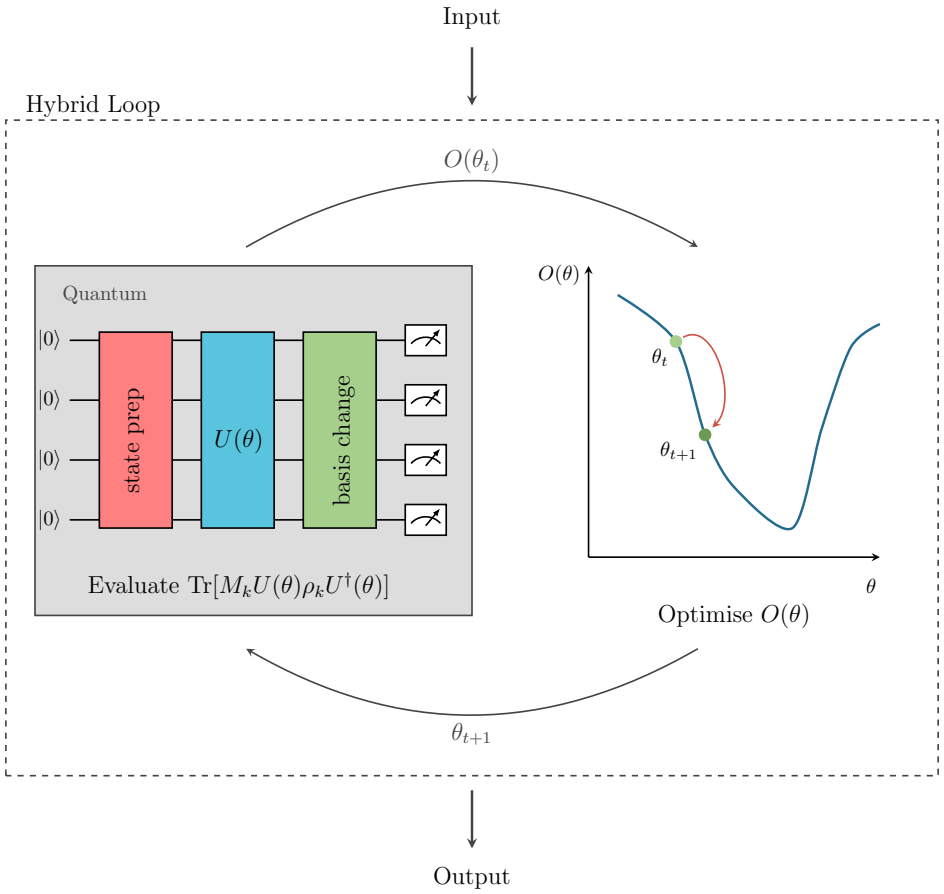


Figure 1.1: **Overview of variational quantum algorithms:** A variational quantum algorithm is a relatively flexible framework which encodes the problem as an objective function $O(\theta)$ parameterised by a set of parameters θ of an ansatz quantum circuit. This ansatz could include sections of the quantum circuit that are not parameterised, for example, some state preparation or basis change prior to measurement which might be based on the input. The problem is solved by iteratively optimising the objective function. At iteration t, the quantum circuit highlighted in grey is evaluated, providing the value of the objective function $O(\theta_t)$. This passes into a classical optimisation algorithm, which proposes a new set of parameters θ_{t+1} to test the objective function, which gets passed back to the quantum device. Once the optimisation has converged, the output is extracted.

prevalence of VQAs in NISQ research there are several challenges in utilising them. Some of these challenges are outlined at the end of this section.

The objective function

The objective function or cost function encodes the problem the VQA is trying to solve. In particular, the objective function is parameterised by a set of parameters θ and outputs some real number. The objective function is designed such that the global minima corresponds to the solution of the problem and is given by the parameters θ^*. Assuming the parameterised quantum circuit representing the trainable ansatz is expressed as $U(\theta)$. In general, there will be some set of inputs, $\{\rho_i\}$ and some observables $\{O_i\}$ that are measured to construct the objective function. Hence, the overall objective function could be represented as

$$O(\theta) = \sum_k f_k(\mathrm{Tr}\Big[O_k U(\theta)\rho_k U^\dagger(\theta)\Big]), \tag{1.2.7}$$

where f_k is some post-processing function that may be classical [25].

The quantum ansatz

The potential advantage of VQA comes when the objective function is not classically simulable. This occurs when the parameterised quantum circuit $U(\theta)$, often referred to as the quantum ansatz, is sufficiently expressive so as not to be classically simulable. However, the choice of ansatz can have various tradeoffs related to trainability. Convergence speeds may be slower with more expressive ansatz, and on NISQ devices, a deeper circuit ansatz is more prone to error. Two common types of ansatz are problem-inspired ansatz and hardware-efficient ansatz [3].

The problem-inspired ansatz constructs the unitary operator as a sequence of time evolutions under a Hamiltonian h_i such that

$$U_i(\theta_i) = e^{-ih_i\theta_i}, \tag{1.2.8}$$

where the parameter θ_i acts like time. In the problem-inspired ansatz, the overall unitary is constructed by a set of operations that are motivated by the problem. A common example of this type of ansatz is the unitary coupled cluster ansatz used in quantum chemistry problems [26]. Problem-inspired ansatzes can benefit over the hardware-

efficient ansatz by using information about the problem to restrict the search space of the variational ansatz. Computationally, problem-inspired ansatz have been shown to converge quicker, with fewer ansatz layers and optimiser iterations, than hardware-efficient ansatz [27]. However, they are often challenging to realise on NISQ hardware due to the restricted connectivity of these devices, the specific gate set they implement and the limited depth they can achieve with good fidelity.

An alternative approach is a hardware-efficient ansatz that prioritises device constraints to construct $U(\theta)$. Generally, the hardware efficient ansatz is constructed from repeatedly applying a subcircuit comprising of a set of two-qubit and single-qubit gates. Each of these subcircuits is called a layer. Assuming a single layer is given by $U_i(\theta_i)$, the full hardware efficient ansatz is given by

$$U(\theta) = \prod_{i=1}^{L} U_i(\theta_i), \tag{1.2.9}$$

where the layers are repeated L times. The exact choice of the layers in the hardware efficient ansatz will depend on the device characteristics, including its topology, gate set and performance. One of the advantages of hardware-efficient ansatz is its flexibility, and it is often applied in the context of problem-agnostic algorithms. In general the expressibility and entangling capability of a hardware-efficient ansatz is controlled by the number of parameters in each layer, and the number of layers applied, and has been studied for a variety of ansatz classes [28, 29]. However, the choice of ansatz will also impact how fast the VQA can converge to its solution, and the hardware-efficient ansatz is often challenging to train.

Optimisation

The main classical computation in VQA comes from the optimisation of the objective function to find an optimal parameter set θ^*. Performing optimisation in the context of NISQ devices is particularly challenging due to the noise present and the scaling of measurements needed with the number of qubits measured. The optimisation strategies fall into two broad classes: gradient-based optimisation and gradient-free optimisation.

One of the most common classical methods for optimising an objective function $O(\theta)$ is using the gradient with respect to the parameters $\theta = (\theta_1, \theta_2, \ldots \theta_N)$. The gradient at a point provides information about the direction of the greatest change of the objective function in param-

eter space at that point. Hence, algorithms such as gradient descent [30] will take a step in the direction of the greatest negative change to attempt to find the global minima. The update rule for gradient descent is as follows

$$\theta_i^{t+1} = \theta^t - \eta \partial_i O(\theta), \tag{1.2.10}$$

where t is the step in the iterative update of θ^t and η is a hyperparameter corresponding to the learning rate. A common strategy for extracting the gradient from a quantum device is the finite difference method and similar variations specific to quantum ansatzes, such as the parameter shift rule [31, 32]. To get accurate computations of the gradients using these methods often requires a large number of measurements.

Gradient-free methods do not rely on gradient information from the quantum device to perform their optimisation. There is a broad range of gradient-free optimisation strategies. Examples include sequential minimal optimisation [33], where the problem of optimising the entire high-dimensional objective function is broken down into smaller problems. In the quantum domain, the Rotosolve and Rotoselect algorithms utilise this approach to optimise parameterised quantum circuits [34]. Alternatively, in situations where evaluating the objective function is expensive, surrogate model-based optimisation may be preferable [35]. In this approach function evaluations from all parameter values are used to construct an approximation of the search space, named the surrogate model. The optimisation itself happens on the surrogate model.

The noisiness of NISQ devices and the sampling times of some quantum devices make resolving an objective function and thus optimising parameterised quantum circuits particularly challenging. Often, optimisation algorithms are chosen that are known to be noise resilient and offer good sampling efficiency. In particular, the simultaneous perturbation stochastic approximation (SPSA) algorithm [36] is a noise-resilient optimiser that does not require explicitly evaluating the gradient on the quantum device. The algorithm approximates the gradient in each iteration step by evaluating the objective function twice. The number of evaluations is independent of the number of parameters, and this optimiser is considered noise resilient; therefore, it is used in many NISQ algorithms.

Challenges

One of the primary challenges of VQAs is the barren plateau problem. This problem arises when the cost function landscape produced by the objective function is very flat. Barren plateaus are known to be caused by a number of factors including overly expressive parameterised quantum circuitss (PQCs) [37], the choice of initial states and measurement [38] and even noise [39]. Recently, the presence of barren-plateaus has been linked to the curse of dimensionality [40]. The consequence of the gradients of the cost function landscape are exponentially vanishing for an arbitrary choice of parameters, meaning gradient-based optimisation requires many samples to determine gradients [41]. It has been shown that even gradient-free optimisation strategies find it challenging to deal with barren plateaus as an exponentially high number of measurements are needed to gain any information from this cost-function landscape. The causes of barren plateaus vary but are often associated with a more expressive quantum ansatz. Using a problem-inspired ansatz or a reduced ansatz may improve the trainability of the VQA. However, this reduced ansatz may not have a solution within its expressible domain.

In addition, working with NISQ devices inevitably means dealing with a large amount of noise. The choice of using VQAs on NISQ devices in this thesis was due to the fact that these algorithms are inherently noise resilient. As the optimisation trains on the NISQ device, the training allows for a parameter set to be chosen that is somewhat adaptable to the noise present on the device. This is particularly true for coherent noise [42], which can be represented mathematically as unitaries and often involves a shift in the desired gate angle, where variational quantum algorithms have been shown to theoretically and experimentally adapt to such noise [43, 44]. However, training parameterised quantum circuits in the presence of noise can be challenging despite the existence of noise-resilient optimisers such as simultaneous perturbation stochastic approximation (SPSA). To counter the noise on a device, a variety of error mitigation strategies have been proposed for NISQ device. The tradeoff of the reduction in noise and the increased cost of running error mitigation needs to be carefully accounted for when designing VQA.

1.3 Representing Quantum Tensor Networks

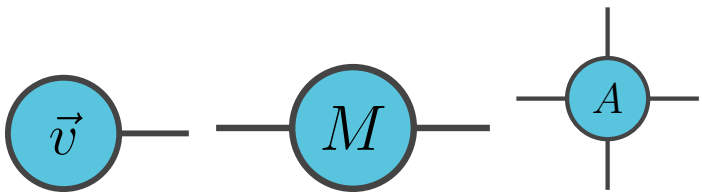
1.3.1 Introduction to Tensor Networks

Tensor networks refer to a broad class of numerical and theoretical techniques predominantly focused on solving problems related to strongly correlated quantum systems and classical statistical mechanics. Describing such systems requires handling large Hilbert spaces that are often impossible to represent and manipulate directly. An overarching theme of tensor networks is defining and manipulating mathematical descriptions of the Hilbert spaces of these systems based on their entanglement structure [45, 46].

One of the most straightforward classes of tensor networks, and a model class of focus in this thesis, is the matrix product state (MPS) [47–49]. The MPS is a simple ansatz class for representing 1D spin systems, with seminal results including the analytical study of AKLT [50, 51] states and consequently translationally invariant finitely correlated states [48]. Numerically, links between MPSs and the density matrix renormalization group (DMRG) algorithm [52, 53] lead to MPSs being the state-of-the-art ansatz when investigating gapped 1D systems [45]. Beyond theory, there exist a number of numerical techniques based on finite size scaling and finite-entanglement scaling which allows MPS techniques to be used to study critical and gapless quantum systems [45, 54–56]. In addition to this, one can use tensor networks to study higher dimensions, such as through the use of projected entangled pair state (PEPS) models to study 2D systems. Models also exist to study gapless and critical systems through multiscale entanglement renormalisation ansatz (MERA) networks [57].

Despite originally being developed within the context of condensed matter physics, tensor networks have proved their utility in a broad range of fields. For example, MERA networks are known to have close links with renormalisation theory and have been used to investigate problems in AdS/CFT [58]. They can be used to understand and investigate several results in quantum information, with tools such as the ZX-calculus having close ties to tensor networks [59]. Classical tensor network algorithms have recently been devised for problems entirely outside quantum physics. This includes tensor networks being used as models in the context of machine learning [60, 61] and solvers for fluid dynamics problems [62].

This thesis focuses on the link between tensor networks and quantum algorithms. For this purpose, MPS based models are advanta-



(a) (b) (c)

Figure 1.2: **Example Tensors:** Examples of tensors in the diagrammatic formalism. The tensor $\vec{v}$ (a) is a vector, M (b) is a matrix and A (c) is an order-4 tensor.

geous as there is a direct translation between canonical MPS and quantum circuits [63, 64]. Though quantum computers have only been available recently to test these ideas, the link between MPSs and sequential generation of quantum states through unitaries has been well understood since the mid-2000s [65–67]. Given the flexibility of MPS and their direct link to quantum circuits, there has been significant recent interest in using tensor networks to design quantum algorithms [61, 63, 68, 69]. This thesis focuses on applying insights from MPS literature to solve problems in quantum simulation and quantum machine learning.

1.3.2 Tensor Network Basics

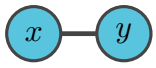
This section introduces the basics of tensor networks necessary for the remainder of this thesis. To begin, we introduce the fundamental primitives needed for working with tensor networks, including the graphical calculus that often simplifies the tensor network notation.

Tensor networks are fundamentally graphs comprised of nodes, referred to as tensors, and edges that define a contraction between the tensors on the edge. A tensor can be viewed as a multi-dimensional array, usually comprised of complex numbers [70]. They are a generalisation of vectors and matrices to more dimensions. The order of a tensor is the number of indices it has. Therefore, a vector is an order-1 tensor, and a matrix is an order-2 tensor [71]. Diagrammatically, a node on a graph draws a tensor of arbitrary order with the same number of edges out of the node as the order of the tensor as shown in Figure 1.2.

Contractions form the foundational routine of tensor network algorithms. If two pairs of indices have an edge connecting them in the tensor network diagram, then a contraction is performed on this index. Mathematically, this is equivalent to setting both these indices to have

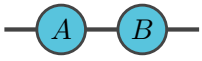
the same value and performing an summation over the repeated index following the Einstein convention. The edge represented by the repeated index is sometimes referred to as the virtual index, and the size of its space is known as the *bond dimension*. Several primitive operations in linear algebra can be written in this form. For example, the inner product between two vectors $\vec{x}$ and $\vec{y}$ is written as

$$\langle \vec{x}, \vec{y} \rangle = \sum_i x_i y_i = \tag{1.3.1}$$



or a matrix multiplication between matrix A and B amounts to

$$AB = \sum_k A_{i,k} B_{k,j} = \tag{1.3.2}$$



[46]. As can be seen, the diagrammatic notation that tensor networks utilise can represent equivalent mathematical expressions much more simply. This is particularly true as the networks become significantly more complex.

Contraction is one of the most frequently used primitives in most tensor network algorithms. Therefore, carefully considering the cost of contracting a tensor network is important. Like many important properties of tensor networks, the cost of contraction scales with the bond dimension χ. Another important factor to consider when contracting an entire network is the order in which the edges are contracted. Different contraction orders can result in vastly different computational costs. For a general tensor network, the question of the optimal contraction sequence is known to be an NP-hard problem. However, for many common networks, such as MPSs, used throughout this thesis, the optimal contraction is relatively easy to find [71].

In addition to combining tensors using contraction, several factorisation routines are used to split tensors. One of the most common routines is the singular value decomposition (SVD), a widely used factorisation throughout mathematics. The SVD takes in an arbitrary complex matrix A with dimension $d_1 \times d_2$ and decomposes it into

$$A = U \Lambda V^\dagger. \tag{1.3.3}$$

Assuming $d_1 \geq d_2$ then U is an isometry of dimension $d_1 \times d_2$, Λ is a diagonal matrix containing the singular values of A of dimension $d_2 \times d_2$ and V is a unitary of dimension $d_2 \times d_2$. For tensors of order > 2, the indices are grouped to form a matrix before passing the entire object into the SVD and ungrouped afterwards as necessary.

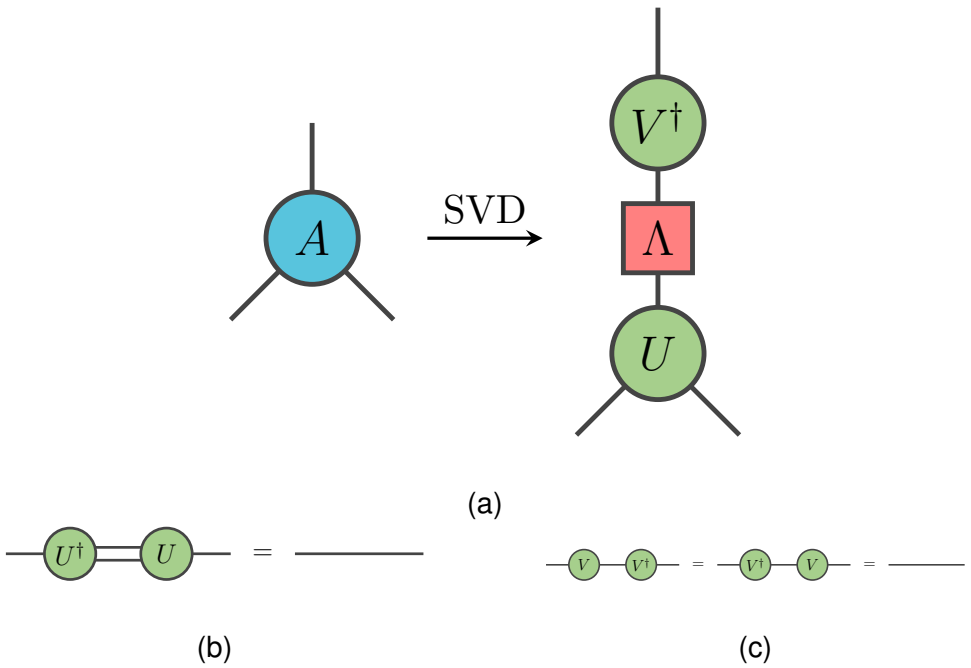


Figure 1.3: **Diagrammatic SVD:** Diagrammatic equations demonstrating the SVD factorisation of a order-3 tensor. The bottom two legs are grouped to form a matrix to apply the SVD to. The tensor U is an isometry containing the bottom legs. It satisfies the isometry condition outlined by b). This tensor connects to a diagonal tensor Λ containing the singular values of A. Finally the tensor $V^\dagger$ represents a unitary operation satisfying c).

Figure 1.3 shows the application of the SVD to an order-3 tensor and shows diagrammatic equations for the isometry conditions $U^\dagger U = I$ and $V^\dagger V = VV^\dagger = I$ where I is the identity.

The SVD is a beneficial routine for the truncation of bond dimension [57]. Truncation refers to approximating a tensor network by reducing the bond dimension of the composite tensors. Ideally, the output of the tensor network is not significantly affected by the reduction of bond dimension. Several truncation techniques can be chosen based on the problem and model being investigated. One general-purpose method is to approximate a tensor by only preserving a subset of the singular values in the SVD. This can be done by preserving a fixed number of the largest singular values or assigning some threshold magnitude for the singular values, below which they are cut off. The exact effect of performing this truncation varies depending on the network, but for quantum states, this is related to performing an approximation whilst preserving the 2-norm or Frobenius norm of a state [48].

The utility of tensor networks comes from their ability to represent high-dimensional Hilbert spaces using a low-dimensional factorisation. For example, consider a standard quantum many-body state. For this

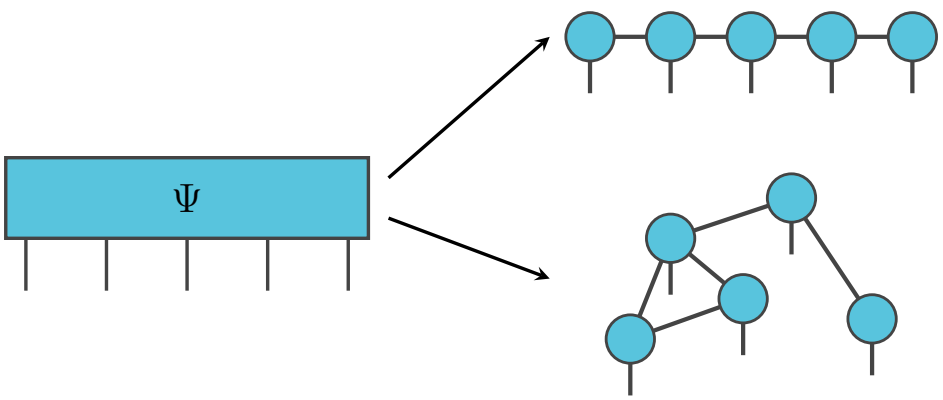


Figure 1.4: **Decomposing quantum state into tensor networks:** Tensor networks can represent states in a high-dimensional Hilbert space such as ψ using a factorisation of low-dimensional tensors.

example, we consider a set of N spin-1/2 systems. Each spin contributes two levels to the overall system. The general wavefunction for the overall system can be written as follows

$$|\psi\rangle = \sum_{i_1,i_2,\ldots,i_N} \Psi_{i_1,i_2,\ldots,i_N} |i_1\rangle \otimes |i_2\rangle \otimes \ldots |i_N\rangle \,, \tag{1.3.4}$$

where $|i_k\rangle$ refers to the local basis for the k-th particle. This equation requires a representation of the object Ψ, which scales exponentially with N, containing 2^N complex numbers. Therefore, directly representing Ψ for any system of reasonable size quickly becomes intractable with increasing system size. The main idea behind tensor networks is to break down this large object Ψ into a set of much smaller objects, usually with a polynomial scaling with system size [70]. Figure 1.4 summarises this idea. Usually, the underlying structure of the network of smaller objects is inspired by information about the entanglement structure of the state we are trying to simulate [45]. An example of this applied to 1D spin systems are discussed in the next section.

1.3.3 The Matrix Product State (MPS)

The matrix product state (MPS) is a class of tensor networks representing 1D systems [47, 53, 70, 72, 73]. They are comprised of a sequence of tensors connected in a chain. There is one tensor per lattice site, and the edges (virtual indices) connecting the tensors have a bond dimension of size D. The open edges are called the physical indices and have dimension d. These indices represent the physical degrees of freedom of the local Hilbert space of the system. Mathe-

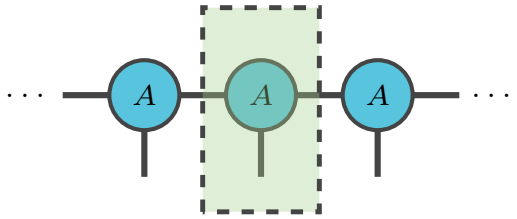
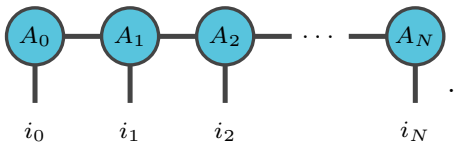


Figure 1.5: **Representing an iMPS:** Representing an translationally invariant matrix product state (iMPS) requires identifying a unit cell and repeating that unit cell. This diagram shows a single-site unit cell highlighted in green.

matically, a finite MPS with open boundary conditions can be written as

$$|\psi\rangle = \sum_{i_1,i_2,\ldots i_N=1}^{d} A_1^{i_1} A_2^{i_1} \ldots A_N^{i_N} \, |i_1 \, i_2 \, \ldots \, i_N\rangle \, . \tag{1.3.5}$$

Diagrammatically this equation is

$|\psi\rangle =$  . (1.3.6)

For periodic boundary conditions, the equation is readily modified to include a trace on the open edges on either end. Furthermore, an translationally invariant matrix product state (iMPS) can be represented by simply repeating the unit cell and forming a chain. For example, a single site unit cell iMPS can be written as shown in Figure 1.5, where the unit cell is highlighted in green.

Given an arbitrary wavefunction $|\psi\rangle$, an MPS can be readily constructed using a sequence of SVDs. To do this, consider an arbitrary wavefunction represented by the tensor $\Psi_{i_1,i_2,\ldots,i_N}$. We first group the legs $i_2 \ldots i_N$ together to form a matrix and perform an SVD such that $\Psi_{i_1,[i_2 \ldots i_N]} = USV^\dagger$, where S contains D eigenvalues. The isometry U becomes first tensor $A_1 = U$ and is connected to the remaining object $\Psi'_{i_2,\ldots i_N} = SV^\dagger$ with a bond of dimension D. This procedure can be applied repeatedly in sweeps to each subsequent Ψ' to produce the MPS [48]. During this process approximate MPS can be generated by truncating the singular values in the SVD. Figure 1.6 outlines the process of generating an MPS using SVD sweeps. Note this procedure is not usually necessary to prepare a MPS. Often the specific routine used will depend on the problem. For example, when preparing groundstates of Hamiltonians an algorithm such as DMRG will be

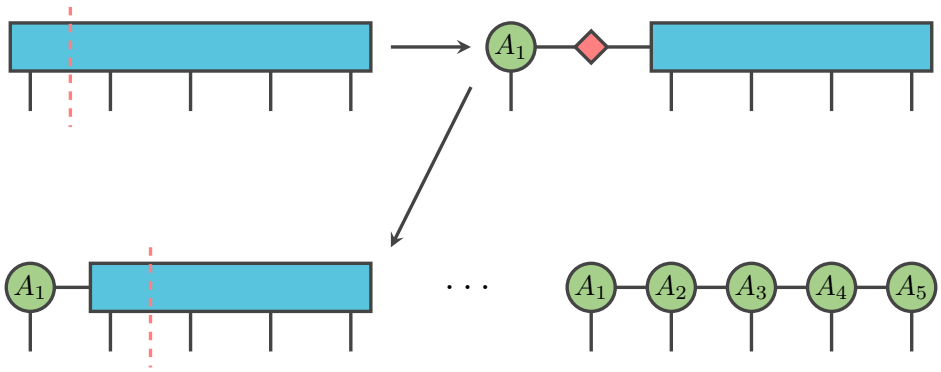
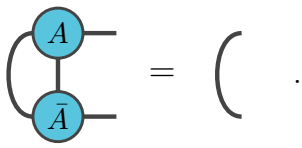


Figure 1.6: **Generation of MPS using SVD:** Generating an MPS from an arbitrary quantum state requires a sequence of SVDs. The physical legs are grouped to include one physical leg on the left, the virtual index on the left, and the remaining physical legs on the right. After applying an SVD the tensor representing the isolated physical leg is formed. The singular values are combined into the remaining tensor. This process is applied repeatedly until the MPS is formed.

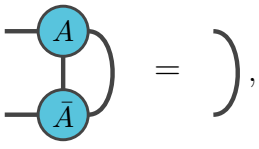
used to prepare the MPS representing the groundstate.

MPS have a gauge invariance that allows an arbitrary MPS to be put in canonical form [48]. Gauge invariance refers to the fact that an arbitrary unitary U can be inserted onto the virtual indices as $U^\dagger U = I$ and absorbed into the tensors without affecting the action of the MPS in the physical space. This is useful as it allows us to transform an arbitrary MPS into useful forms, such as their canonical form. It should be clear from the SVD sweep procedure above that generating an MPS in this way enforces the following isometry condition to each tensor



(1.3.7)

Here the tensor A contracted with its conjugate tensor $\bar{A}$ on the physical and left virtual index is an identity map on the right virtual indices. When all the tensors in an MPS satisfy this property, it is called a left-canonical MPS. Note that performing the SVD sweep in the reverse direction allows for the tensor to satisfy the reverse condition

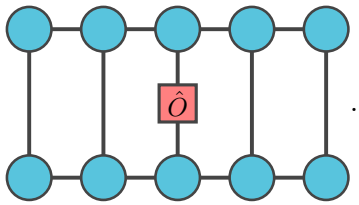


(1.3.8)

and is referred to as the right-canonical form of the MPS. In addition, we can select a site i on an MPS such that all the tensors to the right

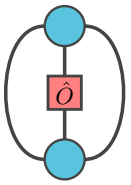
of the site are in right-canonical form and all the sites to the left of this site are in left-canonical form. This process results in a mixed canonical form of the MPS.

The canonical forms are useful as they allow us to greatly reduce the complexity of evaluating tensor network diagrams. For example, consider evaluating the expectation value of a single site operator $\hat{O}$ on site i of an abitrary MPS $|\psi\rangle$. Diagrammatically, this expectation would be written as



. (1.3.9)

The computational complexity of contracting two MPSs together scales with bond dimension D as $\mathcal{O}(D^3)$. As this contraction is the basic operation for calculating quantities such as expectation value or correlation functions, MPS can be used to calculate these quantities efficiently. Furthermore, if the MPS is in mixed canonical form with the central site at site i, then the expectation value highlighted earlier can be written as



, (1.3.10)

thus greatly reducing the cost of computing this quantity further.

The basic properties of MPS have been studied in great detail analytically. In general, MPS can represent any arbitrary quantum state simply by increasing D [70]. MPSs follow an area law in the entanglement entropy where the entanglement entropy of the half-chain scales as $S = \log_2 D$ [74]. As low energy states of gapped quantum systems in 1D often follow an area law [75], this means that MPSs are very efficient at capturing properties of such states with a fixed finite bond dimension. MPS have a finite correlation length as the correlations between two sites decay exponentially with the distance between the sites. This means that MPS have trouble representing critical systems where correlation lengths are known to diverge. However, in 1D, this divergence is polynomial in D, and MPS can still represent these states efficiently [49].

Numerically, MPS are an extremely flexible model class which forms the core for several algorithms in condensed matter physics. For ex-

ample, it is state of the art for performing variational state optimisation, such as the preparation of groundstates in 1D using the density matrix renormalization group (DMRG), that can be formulated in the language of MPS [48]. For performing time evolution, several algorithms exist, including time-evolving block decimation (TEBD) and time dependent variational principle (TDVP) [57]. In recent years, MPSs have been adapted to solve problems in other fields, including working as a model class for machine learning problems [60] and solving partial differential equations in fluid dynamics [62].

1.3.4 Tensor Networks on Quantum Circuits

Quantum circuits can be used to represent isometric tensor networks exactly. This section outlines a method to embed a canonical MPS onto a quantum circuit. This is done through sequential generation using a staircase of unitaries as shown in Figure 1.8 [63, 66].

Consider the tensors of a canonical MPS. We show that for a MPS of bond dimension D and a physical index space of size p can be represented by a sequence of unitaries of size $\log_2(Dp)$. Take for example a left-canonical MPS represented by equation 1.3.5 with the tensor at index j given by $A^{i_j}_{\alpha_j,\alpha_{j+1}}$. Note that we have suppressed the tensor index j in this notation. The index α_i corresponds to the virtual index. The left canonical isometry condition written explicitly states that

$$\sum_{i_k,\alpha_k} (A^{i_k}_{\alpha_k,\alpha_{k+1}})^* A^{i_k}_{\alpha_k,\alpha_{k+1}} = \mathbb{I}_{\alpha'_{k+1},\alpha_{k+1}}, \tag{1.3.11}$$

where $\mathbb{I}$ is the identity map. The tensor A^{i_k} can therefore be viewed as an isometry mapping acting on a larger Hilbert space $|\alpha_k, i_k\rangle$ where the left virtual index and the physical index are grouped. The mapping has an output in the space of the right virtual index $|\alpha_{k+1}\rangle$. An isometry can always be embedded in a unitary acting on some reference state without block encoding. This can be done by finding a set of orthogonal vectors to the ones in the isometry and filling out the remaining space of the unitary. Projecting on the reference state removes this arbitrary set of orthogonal vectors. We can view the tensor as

$$A^{i_k}_{\alpha_k,\alpha_{k+1}} = \langle \alpha_k, i_k | \, U^k \, | \alpha_{k+1}, 0 \rangle \, . \tag{1.3.12}$$

We choose U_k to represent the embedding unitary. The reference state the unitary acts on is the $|0\rangle$ state, the standard initial state prepared on a quantum device. The embedding is summarised in Figure 1.7.

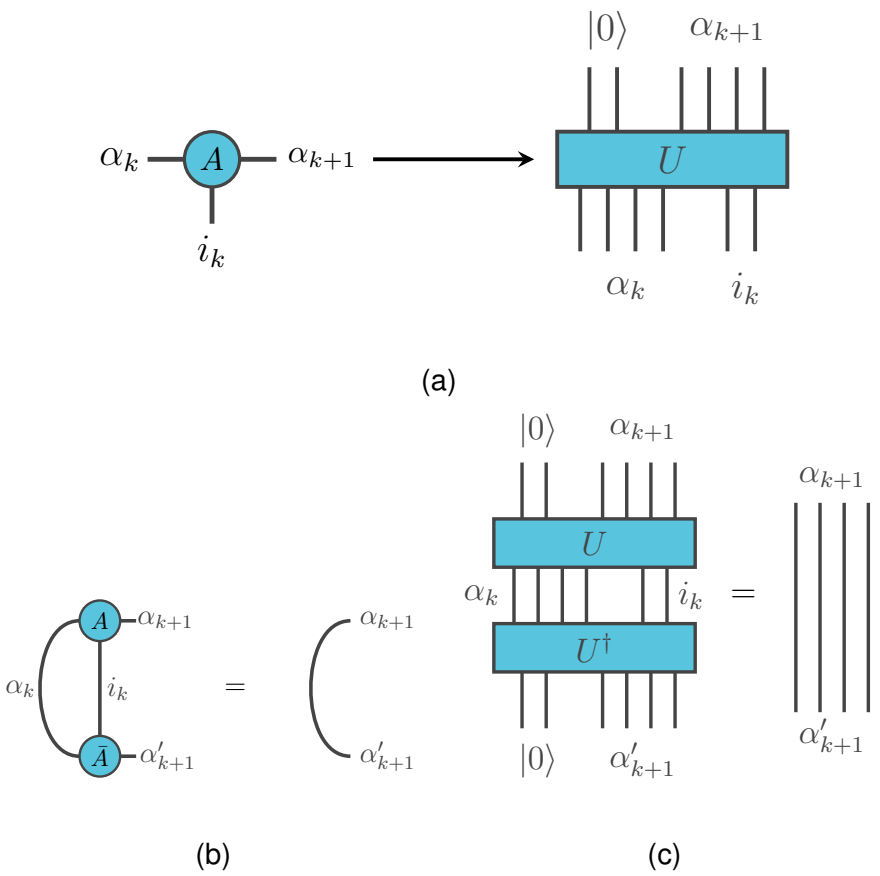


Figure 1.7: **Embedding an MPS tensor in unitary:** Figure a) shows how to embed an arbitrary left canonical tensor into a unitary such that the overall MPS forms a staircase sequential circuit. This embedding satisfies the isometry condition given by b) which maps to the circuit equation c).

The figure shows that this unitary embedding readily satisfies the left canonical condition.

We can now see that the number of qubits necessary to represent the unitary is $n = \log_2(Dp)$. If the tensor A^{i_k} has a bond dimension D and the physical domain of size p, the unitary U^k acts on a space of size Dp. The number of qubits necessary to represent a unitary of this size is $\log_2(Dp)$. Hence, there seems to be an exponential reduction in resource requirements for representing an arbitrary MPS on a quantum device. Connecting these unitaries up to represent the full MPS results in a circuit forming a staircase as shown in Figure 1.8a.

For a spin 1/2 system where $p = 2$, the number of qubits necessary to represent this unitary is $\log_2(D) + 1$. Therefore, an MPS representing a spin 1/2 system with bond dimension 2 is comprised entirely of 2-qubit unitaries. As the exact decomposition of 2 qubit unitaries is possible using the KaK decomposition [76], these MPSs can be represented exactly on a quantum circuit. For a higher bond dimension,

decomposing an arbitrary unitary of size $\log_2(D)$ can eliminate the advantage gained by reducing the resources required to represent the circuit. Approximate decomposition routines have been proposed for such unitaries. One such procedure specific to MPS is to decompose the unitary representing an individual tensor by a reverse MPS with a smaller bond dimension . When applied with a smaller MPS of bond dimension 2, this method corresponds to a brick wall circuit of fixed depth and has been applied to time evolution problems [63].

Assuming an arbitrary bond dimension MPS unitary can be represented on a quantum circuit, the quantum computer is efficient at evaluating valuable quantities. For example, looking again at a single-site operator $\hat{O}$, to evaluate the expectation value, we first decompose the operator into a sum over measurable quantities such as Pauli operators; $\hat{O} = \sum_i \alpha_i \sigma_i$. We then run the circuit, measuring the expectation on each Pauli term σ_i sequentially and summing with the weights α_i to calculate the expectation value. Given a quantum device of sufficient size, these quantities can be measured in parallel, thus resulting in a circuit that scales polynomially with the number of sites and logarithmically with the bond dimension. Similar costs hold true for other quantities, such as correlators and order parameters.

The staircase MPS circuit is not the only way to sequentially generate an MPS on a quantum device. An alternate circuit form is shown in Figure 1.8b for devices with measurement and reuse protocols. The measurements have been moved forward in time, and the physical index space is constantly measured and reset. As opposed to relying on a large number of qubits but a reasonably short coherence time, such as in the staircase layout, this new layout relies on fewer qubits but a longer coherence time. Therefore, we refer to the staircase layout as the *space-like* layout throughout this thesis and the alternate circuit as the *time-like* layout. As the space-like circuit grows in the number of qubits to increase bond dimension or number of sites, the time-like circuit will increase analogously in circuit depth. Therefore, the efficiency in representing these two types of circuit are equivalent, however, they may perform differently based on the hardware. The ability to rewrite MPS in this way to emphasise the potential strengths of a given device dramatically increases the portability of these algorithms between different NISQ architectures.

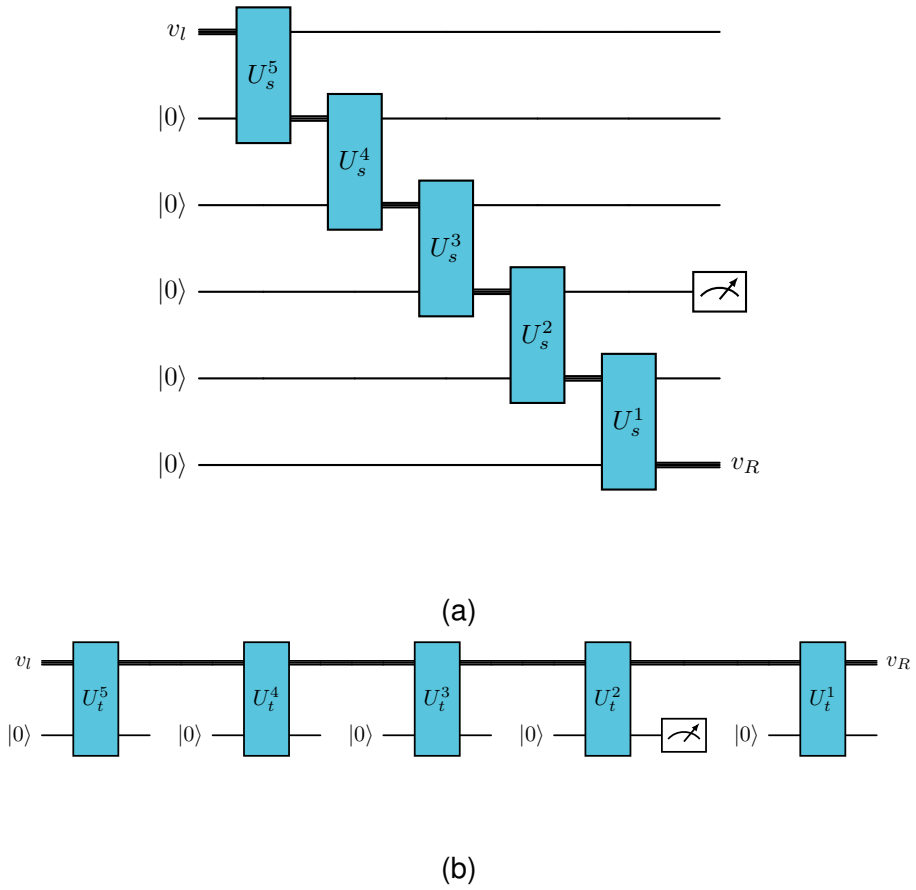


Figure 1.8: **MPS expectation as quantum circuits:** The expectation value of a single site operator on the MPS tensor two as a sequential circuit. The figure a) shows the space-like version of this circuit with a state unitary U_s^i and b) shows the time-like version of the circuit with a state unitary U_t^i. The space-like and time-like unitaries are equivalent up to a $SWAP$ gate. The virtual index is highlighted by the thick wire.

Looking Forward

The next section of this thesis extends the study of the quantum MPS ansatz for the purposes of quantum simulation. In particular, I adapt the staircase MPS representation to perform groundstate optimsation. Unlike prior algorithms which utilise this ansatz [68], this algorithm allows for the preparation of arbitrary states without analytically finding the parameterisation and gate angles required to represent a state. I test the feasibility of this algorithm on Google's Sycamore processor and subsequently develop a novel time evolution algorithm inspired by tensor network techniques.

Part I

Quantum Tensor Networks for Quantum Simulation

Chapter 2

Ground state optimisation

The variational principle is a powerful tool for studying the ground state behaviour of strongly correlated systems. This chapter uses the variational principle with a translationally invariant MPS (iMPS) circuit ansatz to prepare ground states for a quantum system in the thermodynamic limit. This method requires representing and optimising the local expectation values of iMPS circuits using finite circuits. We present an algorithm that can perform this optimisation whilst considering the restrictions of near-term quantum devices. Finally, we present some results from applying this ground state optimisation algorithm on Google's Sycamore architecture to study the ground state properties of the transverse field Ising model across a quantum critical point.

This work was done in collaboration with Andrew Green, James Dborin, Fergus Barratt, Eric Ostby and Thomas O'Brien. My contributions include contributing to the code to run and analyse the results from this algorithm on Google's device. I was also involved in discussions on adapting the algorithm based on the device performance of Sycamore architecture. This work and its results are published and form part of Ref [77].

2.1 Ground state optimisation algorithm

2.1.1 Variational ground state optimisation

A general problem of interest in condensed matter physics is preparing and calculating ground state properties. Many body physics is generally concerned with macroscopic features of the systems where the size of the system N could be of order 10^{23} [78]. Though systems of this size may not be directly simulatable, we can calculate the properties of these systems by taking $N \to \infty$; thereby taking the thermodynamic limit. Tensor networks provide a class-leading technique to calculate the ground state properties of various quantum systems. The MPS is known to exactly represent ground states of gapped 1D local Hamiltonian [79].

Like many numerical techniques, tensor networks calculate ground states using the variational principle. Given a Hamiltonian $\hat{H}$, the variational principle states that finding the ground state ψ_0 of H amounts to minimising the expectation of the energy given by

$$E = \frac{\langle\psi|\,\hat{H}\,|\psi\rangle}{\langle\psi|\psi\rangle}. \tag{2.1.1}$$

Tensor networks restrict the variational class for the variational principle. Hence, the state optimisation occurs within the manifold of the specific tensor network state. For example, a foundational algorithm in tensor networks is the density matrix renormalization group (DMRG) algorithm [48]. Here, the optimisation occurs over the set of MPS with a restricted bond dimension of D. Optimisation occurs iteratively by updating a single site (or two) whilst keeping the remaining tensors constant.

For this work, we focus on the translationally invariant matrix product state (iMPS) ansatz class. Using this ansatz requires careful consideration of the behaviour of states in the thermodynamic limit. The energy diverges with system size, so we need to optimise energy density in the thermodynamic limit. Many classical methods optimise energy density using the iMPS ansatz either indirectly or directly. Infinite DMRG (iDMRG) methods involve optimising the central site and growing the chain repeatedly until the central site converges, usually verified by solving some fixed point equation. Recently, tangent space methods, such as the variational uniform matrix product states (VUMPS) algorithm [80], have been developed which directly optimises the global iMPS state by minimising the energy density directly. Alter-

natively, one can prepare ground states using imaginary time evolution methods such as the infinite time-evolving block decimation (iTEBD) algorithm [45, 57].

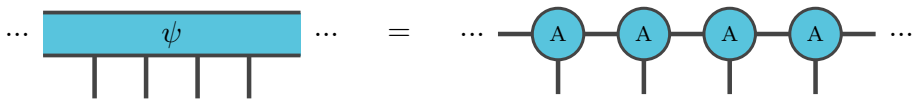
We outline a quantum algorithm to perform ground state optimisation using a quantum circuit ansatz analogous to the classical iMPS. This algorithm - in the spirit of classical algorithms like VUMPS - performs global updates on the quantum iMPS circuit ansatz to minimise the energy density. This section outlines the core aspects of the algorithm for performing ground state optimisation. Initially, we present the circuits required to represent iMPS states on quantum computers. Following this, we outline methods to calculate and optimise the expectation value of local observables, such as energy density.

2.1.2 Local observables of iMPS circuits

Translationally invariant 1D systems can be regarded as homogenous when considering a large enough unit cell, therefore the iMPS in the thermodynamic limit is a chain of uniform state tensors. Hence, a translationally invariant d level spin system is comprised of local sites with an index n and a basis $\{|s\rangle_n, s = 1, \ldots, d\}$ and a total Hilbert space $|\mathbf{s}\rangle = \otimes_n s_n$ can be approximated by an iMPS as

$$|\psi\rangle = \sum_s \prod_n A^{s_n} |\mathbf{s}\rangle \,, \tag{2.1.2}$$

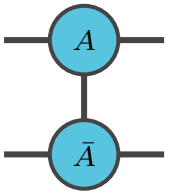
in graphical tensor network notation



, (2.1.3)

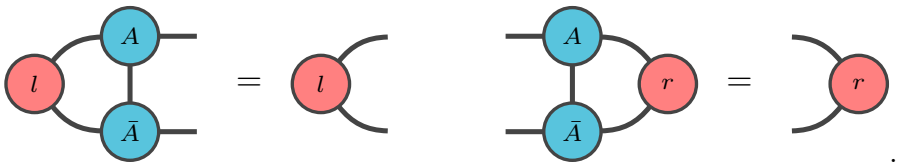
Note for the remainder of this section, we consider spin $1/2$ systems where $d = 2$ and limit ourselves to the graphical tensor network notation where possible.

A core object when calculating properties of iMPS is the state transfer matrix given by

$E =$ 

. (2.1.4)

The transfer matrix can be viewed as a $D \times D$ dimensional operator in the virtual index space. Numerically, we can calculate the leading order eigenvalue η of the transfer matrix and the left and right fixed points l and r of the eigenvalue equations which satisfy



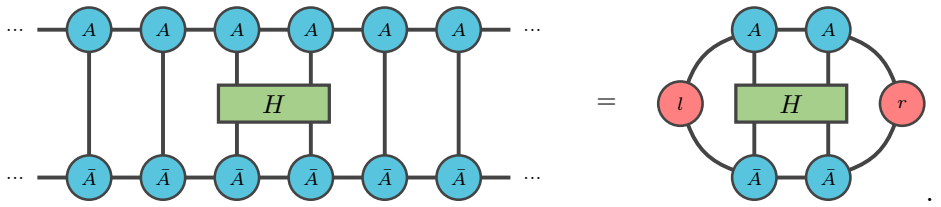
(2.1.5)

To properly normalise the iMPS, we require that the tensors A are rescaled such that $A \to A/\sqrt{\eta}$ and the fixed points l and r are rescaled such that $Tr(lr) = 1$.

The importance of the transfer matrix becomes clear when calculating the energy density. For example consider a hamiltonian H made up of a sum of uniform nearest neighbour interactions $h_{i,i+1}$ such that

$$H = \sum_i h_{i,i+1}. \tag{2.1.6}$$

Calculating the energy density amounts to calculating the expectation value of h; a local expectation on two sites. Everything to the left and right of the local expectation looks like repeated applications of E and hence can be replaced by l and r. Diagrammatically, this is represented as

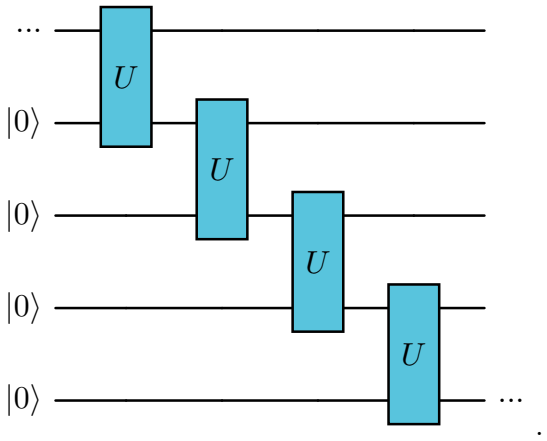


(2.1.7)

In effect, the left and right fixed points, alternatively referred to as environments, encapsulate the behaviour of the infinite state outside of the action of the operator. Hence, to extend finite MPS quantum circuits to the infinite case require finding circuit representations of the environments.

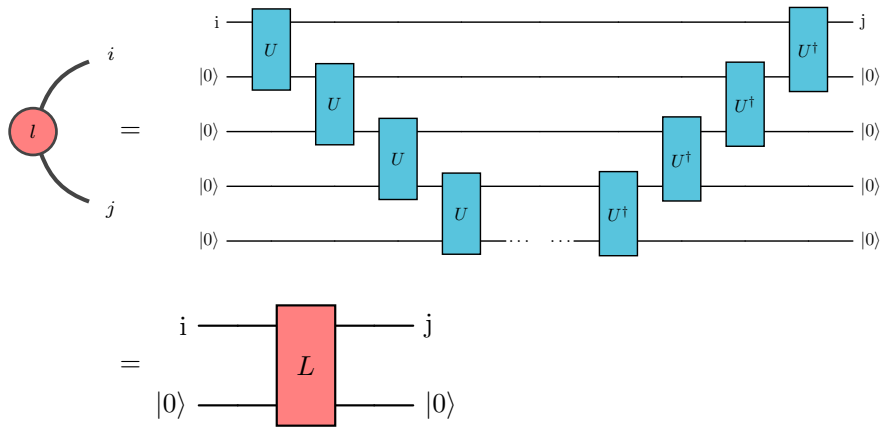
To construct the circuit environment, consider an analogous quantum iMPS circuit where we are concerned with observables on the central two sites. As outlined in the background, we can represent the

iMPS state as a staircase circuit, diagrammatically given by

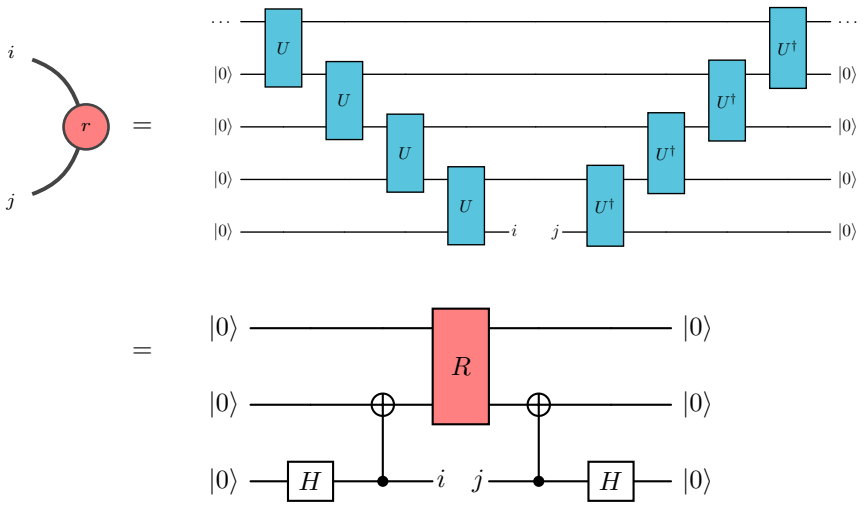


(2.1.8)

When calculating local observables, the infinite staircase to the left and right of the observable can be represented exactly by environments l and r embedded as the column of unitaries L and R as shown by the equations



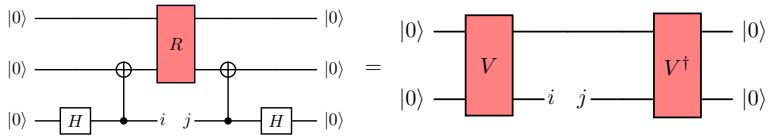
(2.1.9)



(2.1.10)

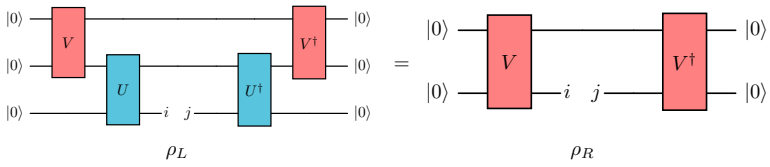
Note that for this discussion, we will represent the circuits in the space-like form for simplicity but equivalent circuits can be drawn in the time-like form.

In practice, to embed the state we put it in left canonical form, simplifying one of the environments. In this form, the environment l is the identity; hence, the environment unitary L resolves to the identity. In addition, we can resolve the right environment unitary R. To do so, we recognise that the right environment is Hermitian and so has a decomposition $r = U\Lambda U^\dagger$ and we can subsequently embed this U in a unitary V such that we can rewrite R as



. (2.1.11)

Therefore, we can write a fixed point equation to resolve for V



. (2.1.12)

Iteratively applying the fixed point equation allows us to construct an efficient representation of the right environment as we expect this iterative application to converge exponentially.

The local expectation values of iMPS circuits are constructed by measuring local observables on finite circuits using these environments. The observable of interest is broken down into sums over Pauli strings on local sites. The expectation of each Pauli string term can be calculated using the circuits shown in Figure 2.1. These circuits show calculations of a 3-site Pauli string in space-like and time-like representations. Note that the measurements shown in these diagrams include a potential Pauli rotation to put the measurement in the correct basis for the Pauli string term.

Calculating local observables of iMPS quantum circuits is a powerful tool in itself. Doing so has allowed for the measurement of a topological phase transition using the space-like layout [68] and probing chaotic dynamics in the time-like layout [69].

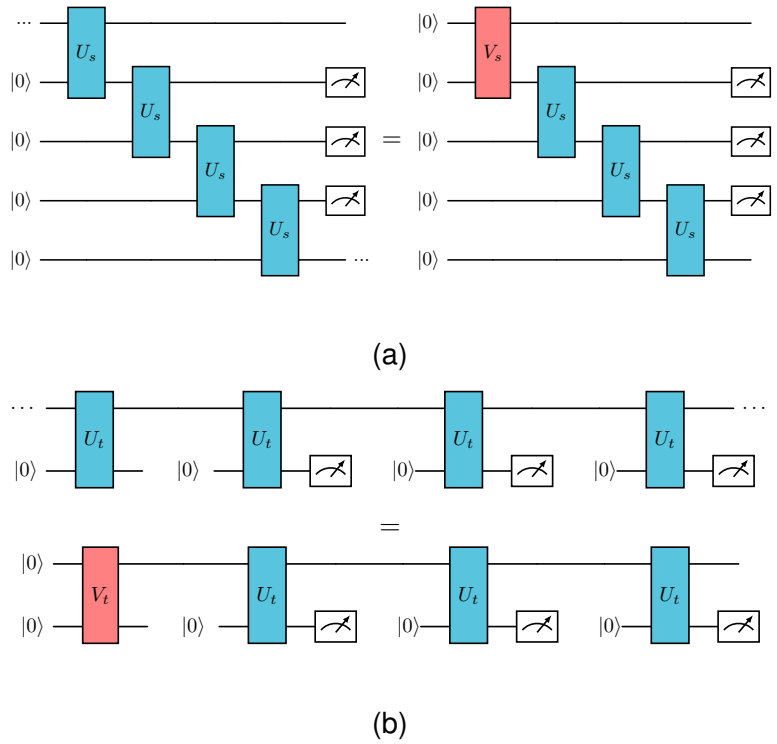


Figure 2.1: **Circuits for expectation of iMPS states:** The circuits shown here calculate the expectation value of a 3-site local observable from an iMPS state using finite circuits in the a) space-like and b) time-like layout. The state unitaries are denoted by a) U_s and b) U_t and the environments as a) V_s and V_t. Note these state unitaries only differ by a $SWAP$ gate at the end.

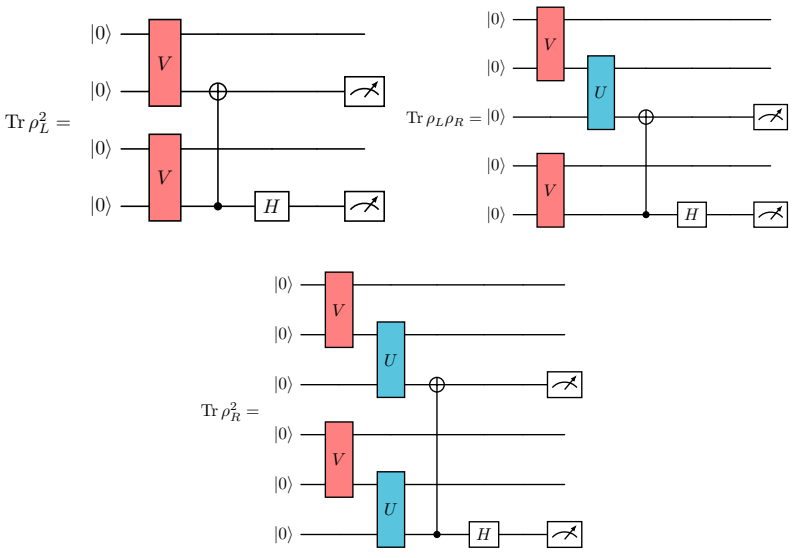
2.1.3 The fidelity density cost function

Given that we now have a way of representing and calculating local observables of iMPS circuits, we move on to performing ground state optimisation using the variational principle. To do so, we outline a variational quantum algorithm that uses the iMPS circuit as a variational ansatz and derives a suitable cost function based on this ansatz. The local observable circuits outlined in the previous section provide a way of constructing cost function terms that minimise energy density.

Previously, we outlined an iterative method to calculate the environment unitary V. However, we would like a cost function term that solves the fixed point equation 2.1.12 through variational optimisation. A trivial term such as the overlap $\mathrm{Tr}(\rho_l \rho_r)$ would not work as the terms on the left and right sides are reduced density matrices. Hence, when $\rho_l = \rho_r = \rho$ this term would amount to optimising the the purity of ρ which is not the correct behaviour. An alternative cost function term to solve the fixed point equation is to minimise the trace distance $\mathrm{Tr}\left[(\rho_l - \rho_r)^2\right]$. Expanding out the trace distance cost function, we find

$$\mathrm{Tr}\left[(\rho_l - \rho_r)^2\right] = \mathrm{Tr}\left(\rho_l^2\right) + \mathrm{Tr}\left(\rho_r^2\right) - 2\,\mathrm{Tr}(\rho_l \rho_r). \tag{2.1.13}$$

The trace distance cost function terms can be calculated on the quantum device using the non-destructive SWAP test [81]. The circuits for these terms are given below



.

Therefore, to perform the ground state optimisation, we choose to optimise the energy density and the trace distance cost function for the environment simultaneously in a combined *fidelity density cost function*

$$C = \langle H \rangle + \mathrm{Tr}\left[(\rho_l - \rho_r)^2\right]. \tag{2.1.14}$$

2.2 Ground states on Sycamore

2.2.1 Model Choice - The transverse field Ising model

We apply the ground state optimisation algorithm using the iMPS circuit ansatz to examine the transverse field Ising model (TFIM). The TFIM Hamiltonian H is given by

$$H = \sum_i \sigma_i^z \sigma_{i+1}^z + g\sigma_i^x, \tag{2.2.1}$$

where g is the strength of the transverse field and σ_i^α is the Pauli matrix $\alpha \in \{X, Y, Z\}$ acting on index i.

The ground state behaviour of the TFIM can be analytically solved using the Jordan-Wigner transformation [82]. Given that we know the analytic behaviour of this model, there is a known ground state quantum phase transition when $g = 1$. In addition, classical MPS algorithms capture the ground state behaviour of this model at low bond order despite there being a critical point. Therefore, this model is a good initial test for the quantum ground state optimisation algorithm run on NISQ devices as the results can be readily compared to analytic and classical numerical methods.

2.2.2 Adapting to Sycamore

We ran the ground state optimisation algorithm on Google's Rainbow device - one of two devices using the Sycamore architecture. The larger version of this architecture was used in Ref. [22] which was comprised of 54 superconducting transmon qubits running in a two-dimensional grid layout with nearest-neighbour coupling with median qubit frequencies at readout of 5.750MHz. Rainbow was a smaller device comprised of 23 qubits. The benefits of this platform are the relatively high clock speeds and readout times with typical repetition rates of around 1-5kHz. However, this device has high error rates, particularly decoherence errors, with T_1 times at idle frequency of 15.54μs [22]. These errors require significant development of error mitigation strategies.

The characteristics of Rainbow lends itself well to the space-like representation of the iMPS circuits. Figure 2.2 shows the structure of the circuits used to represent the terms in the cost function equation 2.1.14 and its layout. These circuits represent a iMPS of bond dimension $D = 2$, where the 2 qubit state unitaries U are highlighted in blue

and the 2 qubit environment unitary V are highlighted in red.

Both the state unitaries U and the environment unitaries V have a restricted parameterisation, with only four parameters in each unitary. Though the restricted parameterisation does not allow for the representation of an arbitrary iMPS, it significantly reduces the costs associated with the optimisation and limits the effects of issues such as barren plateaus.

Calculating the full cost function for ground state optimisation requires the parallel calculation of each term. The circuits in figure 2.2a and 2.2b measure the energy density of the iMPS state by measuring each Pauli term in 2.2.1 independently and the sum is calculated classically. Figure 2.2c and 2.2d calculate the fixed point term by measuring each term in 2.1.13 using the SWAP test. Note that due to the size of Rainbow (with 23 qubits in the grid layout), all of the circuits cannot fit on the device at one time. Therefore, the circuits are laid out such that the three energy terms are measured simultaneously and then the three fixed point terms are measured simultaneously. Figures 2.2b and 2.2d show example layouts of these runs. Experimentally, we used Google's performance metrics, which were updated on a daily basis, to adapt the layout based on the best performing qubits and connectivity.

2.2.3 Error mitigation

Running any algorithm on a NISQ device requires careful consideration of errors. Several common techniques are utilised to account for errors from these devices without running total error correction. These techniques are often broadly classified as error mitigation. We consider three error mitigation techniques when working with Sycamore.

Firstly, near-term devices often have readout errors, where the actual output of the device is not what is measured. Often, the causes of readout error [42] are bit flips during measurement and can be resolved using a confusion matrix. To do so, we run circuits on the device with known outputs, allowing us to build a matrix of transition probabilities A between the ideal measurements M_{ideal} and the measurements from the device M_{noisy}. This matrix of transition probabilities A is the confusion matrix. To reduce the readout error, we invert the confusion matrix and apply it to any measurements from the device as outlined by

$$M_{\text{ideal}} = A^{-1} M_{\text{noisy}}. \tag{2.2.2}$$

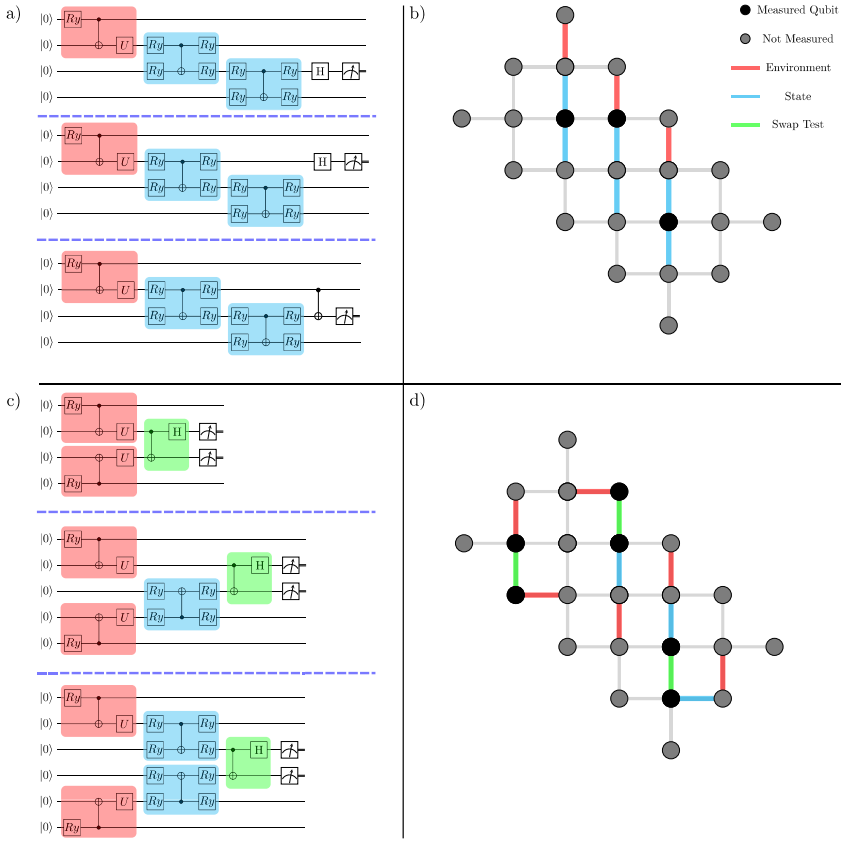


Figure 2.2: **iMPS circuits used to perform ground state optimisation.** a) Circuits representing the three terms in the energy density of transverse field Ising model Hamiltonian. The state unitaries U are highlighted in blue, and the environment unitaries V are highlighted in red. A reduced parameterisation is used with 8 parameters in total, 4 in U and 4 in V. The R_y gate is a parameterised Pauli-Y rotation gate with a single parameter. The W gate is an arbitrary Pauli rotation with three variational parameters. b) An example layout for the energy density circuit terms. The optimal layout was chosen based on the performance of the qubits every day on Rainbow. c) The circuit terms representing the trace distance between the left and right-hand terms of the fixed point equation for the environment unitary. The region highlighted in green is the circuit elements required to perform the SWAP test. d) An example layout of the circuit terms representing the trace distance. This figure forms part of the published work in Ref [77].

This method is not scalable as the size of the confusion matrix increases exponentially with the number of qubits measured. Fortunately, we found that as we were measuring the probability of a particular bitstring, applying the confusion matrix did not significantly change the results.

Daily variations in the performance of the Sycamore processor were common. Due to limitations in the control systems of NISQ devices and variations in fabrication processes, even individual qubit performance can vary significantly. To account for this, we reviewed calibration metrics run on the device daily and carefully selected the qubits and layout of the circuit for each experiment. We used the estimates of single qubit gate error that Sycamore provides to choose the qubits. In particular, we used the single qubit T_1 [83] times and readout errors that are estimated using randomised benchmarking. This process mostly allowed us to eliminate qubits with particularly poor performance. Given that the circuits we run for have a 1D chain structure, we utilise the two-qubit calibration metrics to construct chains with optimal performance on the device. Sycamore provides two-qubit gate errors using cross-entropy benchmarking, and we used the Pauli errors between connected qubits to build the chain. In addition, running multiple copies of the circuit in parallel in a process known as qubit averaging can also account for uncorrelated variations in device performance. Due to the limited size of the device, multiple copies of the ground state optimisation algorithm cannot be run in parallel. However, changing qubits and layouts of the circuit between runs achieves a similar effect. Having access to the actual hardware allowed us to develop protocols which were the most effective at managing errors on the device. We found that these qubit selection and qubit averaging techniques were essential when running on Sycamore, as specific qubits often had anomalously bad performance. Despite the theoretical algorithm being relatively hardware agnostic, practically running on a near-term device requires tuning to it's particular error profile.

Depolarisation error is a significant source of error in many NISQ architectures, particularly on Sycamore. This error is mathematically described when the states of individual qubits probabilistically mix with the completely mixed state. This leads to a shift in the measured energy $\left\langle \tilde{E} \right\rangle$. A common technique for accounting for depolarising error is to shift or rescale the outputs of the circuits based on some known reference value. In the case of ground state optimisation, we parameterise the circuit to a known energy $\langle E \rangle$. Following this, we run the

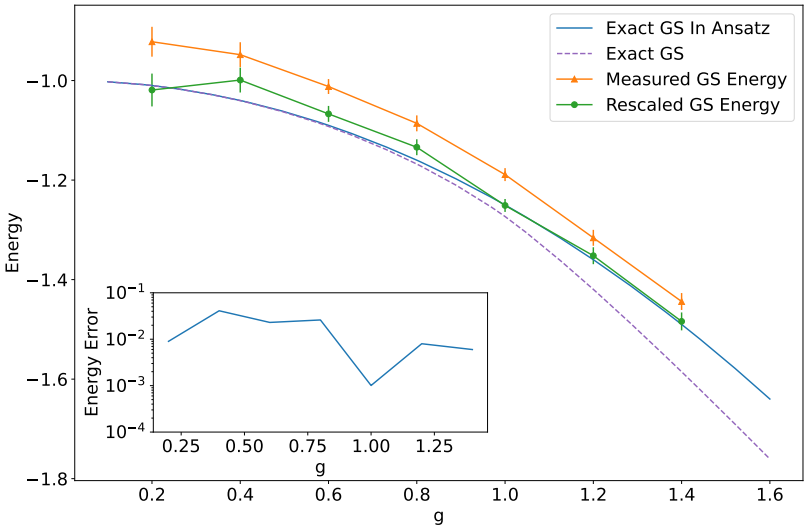
circuit on the device and measure the output energy given by $\left\langle \tilde{E} \right\rangle$. Measuring on the device provides an upper bound on the energy of the prepared state. The rescaling parameter for the remaining circuits of this structure is therefore $\langle E \rangle / \left\langle \tilde{E} \right\rangle$. We calculate the reference value using the transverse field Ising model (TFIM) where $g = 0$, thus producing a ground state energy of -1. The parameterisation of the ground state using our MPS circuits requires setting all the variational parameters close to 0. We measure the output energy of this circuit and use this rescaling parameter for all values of g.

2.2.4 Ground state results

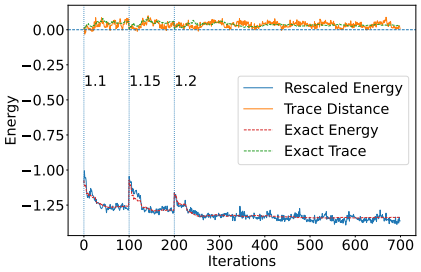
We optimised the combined cost function using Rainbow; Figure 2.3 shows the outcome of this experiment. To do the optimisation on Rainbow, we used the simultaneous perturbation stochastic approximation (SPSA) algorithm - a classical optimisation algorithm known to be noise resilient. The results show that this approach worked well within the phases on either side of the critical point.

When approaching the ground state phase transition, we gradually tuned the transverse field term in the Hamiltonian to reach the critical point in a quasi-adiabatic manner. This involved using the optimal parameters from previous values of g to initialise the optimisation for the next value as we approach $g = 1$. Figure 2.3b shows an example optimisation curve using this quasi-adiabatic approach.

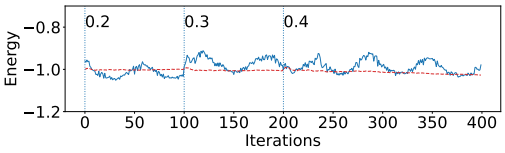
Given this approach and correctly rescaling to account for the depolarisation errors, the results from Sycamore match very closely with the exact value within the ansatz we used. Note that the deviation from the exact in ansatz curve with the analytically exact curve comes from the fact that we are using a reduced parameterisation of our state and environment unitaries. Though this limitation is required given the performance of Rainbow, with subsequent improvements in quantum devices, the reduced parameterisation can be generalised. Despite this, the performance of this algorithm is good, even near the critical point of $g = 1$. The point of most significant deviation from the exact in ansatz results is at $g = 0.4$. This performance can be explained by Figure 2.3c, which shows a typical optimisation curve for this point. There are periodic oscillations in the optimisation curve on the device with a period of about half an hour. Though this oscillation was present throughout the experiment, it seemed to be particularly bad for the $g = 0.4$ curve, thus likely preventing the optimiser from performing correctly. The source of this oscillation is likely a hardware issue.



(a)



(b)



(c)

Figure 2.3: **Results from ground state optimisation on Sycamore:** a)The calculated value of energy at each value of the transverse field g. The dashed purple curve gives the analytic results calculated in the thermodynamic limit. In this experiment, the blue curve gives the result from classically optimising the ground state using the reduced parameterisation for the bond dimension 2 iMPS ansatz. The deviation between the ansatz and the analytically exact curves at large g is due to the reduced parameterisation. The orange and green curves show the measured ground state energy from optimising the iMPS circuit ansatz on Rainbow, both without and with the rescaling. The inset shows the deviation between the rescaled energy measured on Rainbow and the exact energy in ansatz. The largest deviation is found at $g = 0.4$ with an error of about 2.2%. In b) we demonstrate the quasi-adiabatic preparation of the $g = 1.2$ state. Figure c) shows a typical optimisation curve for $g = 0.4$, demonstrating a systematic oscillation during the optimisation that likely caused the larger error at this value.

2.3 Discussion

This chapter outlined a method to variationally prepare iMPS circuits representing the ground states of quantum systems in the thermodynamic limit. To do so, we represent the infinite states on finite quantum circuits to measure local observables using environment unitaries to capture the effects of the infinite states. Subsequently, we outline a cost function for a variational quantum algorithm that allows you to simultaneously optimise energy density and the environment. We implement this ground state optimisation algorithm on Google's Sycamore architecture with appropriate error mitigation strategies. Having access to a real device allows us to investigate the error mitigation strategies and optimisation protocols that are actually required for the algorithm to have good performance. We find that Loschmidt rescaling is a particularly effective strategy against decoherence errors prevalent on Sycamore. We can reproduce the ground state properties for the transverse field Ising model across the quantum critical point, demonstrating the feasibility of this algorithm on near-term quantum devices.

Whether this algorithm provides a potential route to quantum advantage remains an open question. As outlined in the background, there is a route to quantum advantage in the contraction of MPS circuits with high bond dimension on a quantum computer. However, consideration has to be made for the added complexity of representing and optimising states with high bond dimension. It is unclear whether there is a generic way to parameterise high bond dimension iMPS circuit states and environments such that the circuit depths are shallow and the optimisation avoids barren plateaus.

Near-term quantum algorithms must be adaptable to devices with significantly different characteristics due to variations in the underlying architecture. MPS circuits with space-like and time-like layouts greatly enhance their portability. Sycamore utilises superconducting qubit technology and is characterised by relatively short coherence times and high qubit numbers. This architecture feature means it is particularly well suited to the space-like layout of the circuits. Alternatively, many devices, such as trapped-ion architectures, have fewer qubits but much longer coherence times and mid-circuit measurements. The time-like layout of the MPS circuits are more suitable for these devices. It would be helpful to systematically study the tradeoffs between these layouts with different device characteristics to quantify this algorithm's portability further.

Investigating increasing bond dimension using iMPS circuit is likely

a fruitful direction for further study. Optimising the transverse field Ising model does not produce an appreciable difference when scaling the bond dimension from $D = 2$ to $D = 4$. However, other models of interest in condensed matter physics, such as the Heisenberg model, need larger bond dimensions for accurate results. A core requirement for achieving this scaling is improving device fidelity and error mitigation techniques. However, as quantum devices develop rapidly, these tensor network circuits provide a systematic way of increasing the complexity of simulation towards problems of practical utility.

Chapter 3

Quantum state evolution

Simulating the Hamiltonian time dynamics of quantum systems is another route to the utility of near-term quantum hardware. Trotter evolution of quantum matrix product state circuits could provide access to physically relevant states with high entanglement but relatively low complexity [63] that can be difficult to simulate directly classically. Here we outline an algorithm that uses the iMPS ansatz, introduced in the previous chapter, to perform time evolution. This protocol develops the variational quantum algorithm proposed in Ref [84]. Here, we define a fidelity density cost function to perform time evolution inspired by classical MPS time evolution methods. We apply this to two current-generation quantum computing architectures to investigate the dynamical quantum phase transition (DQPT) of the transverse field Ising model (TFIM). Adapting the algorithm to current-generation devices requires careful consideration of sources of error and developing novel techniques to perform optimisation. As a result, we present an architecture-aware and tunable algorithm to run on superconducting and trapped-ion architectures.

This work was done in collaboration with Andrew Green, James Dborin, Lesley Gover and Fergus Barratt. In addition, running on quantum hardware required collaboration with teams from both Google Quantum AI and Quantinuum. My contributions included discussing and developing the quantum algorithm and contributing to the code for the devices. Some of this work and its results are published and form part of Ref [77].

3.1 Dynamics with iMPS circuits

3.1.1 Classical MPS time evolution

The time evolution algorithm outlined in this chapter is inspired by classical techniques used to evolve MPSs. In this section, we review two key classical MPS time evolution algorithms, namely time-evolving block decimation (TEBD) [85, 86] and the time dependent variational principle (TDVP) [87, 88]. Both these algorithms have features that can be used to understand our quantum time evolution algorithm. For a complete review of modern time evolution methods using the MPS ansatz, refer to Ref [89].

Time-evolving block decimation (TEBD)

To time evolve a state $|\psi(0)\rangle$ for some time T, we can consider discretising the evolution into N steps of size $dt = T/N$. This discretisation requires finding a good approximation for the incremental evolution $U(dt) = e^{-iHdt}$. This evolution is subsequently applied to the state at time t ($|\psi(t)\rangle$) so as to prepare the state at time $t + dt$ ($|\psi(t + dt)\rangle$). In TEBD, the state is parameterised as an MPS, and we apply a time evolution operator to it approximated using a Trotterisation of $U(dt)$. Trotterised evolution is unitary and thus decomposable into quantum gates.

The Trotterisation of nearest neighbour Hamiltonians - such as the TFIM - in 1D can be constructed as follows. The Hamiltonian H can is decomposed as

$$H = \sum_i \hat{h}_{i,i+1}, \tag{3.1.1}$$

where $\hat{h}_{i,i+1}$ is the component of the Hamiltonian acting on adjacent sites i and $i + 1$. The exact time evolution operator $U(dt)$ is given by

$$U(dt) = e^{-i\hat{H}dt} = e^{-i(H_{\text{even}}+H_{\text{odd}})dt}. \tag{3.1.2}$$

Here, we note that the Hamiltonian H can be broken down into the sum of two segments. The first segment consists of a Hamiltonian representing all the even lattice sites - $H_{\text{even}} = \sum_i \hat{h}_{i,i+1}$ where i is even. The second segment represents all the odd lattice sites - $H_{\text{odd}} = \sum_i \hat{h}_{i,i+1}$ where i is odd.

Using the Baker-Cambell-Hausdroff formula, we can approximate the sum in the exponential as

$$\begin{aligned} U(dt) &= e^{-i(\hat{H}_{\text{even}}+\hat{H}_{\text{odd}})dt} \\ &= e^{-i\hat{H}_{\text{even}}dt}e^{-i\hat{H}_{\text{odd}}dt}e^{-i[\hat{H}_{\text{even}},\hat{H}_{\text{odd}}]dt} \\ &\approx e^{-i\hat{H}_{\text{even}}dt}e^{-i\hat{H}_{\text{odd}}dt} \equiv U_{\text{TEBD1}} \end{aligned} \tag{3.1.3}$$

This construction gives the first order Trotter decomposition $U_{\text{TEBD1}}(dt)$. Note that $e^{-i\hat{H}_{\text{even}}dt}$ and $e^{-i\hat{H}_{\text{odd}}dt}$ are straightforward to construct as each of the individual terms in h_{even} and h_{odd} commute with each other.

To see why this is a first-order Trotter expansion, consider the expansion of the commutator term in dt

$$e^{-i[\hat{H}_{\text{even}},\hat{H}_{\text{odd}}]dt} \approx \mathbf{1} - i(dt)^2[H_{\text{odd}}, H_{\text{even}}]. \tag{3.1.4}$$

Therefore $U(dt) = U_{\text{TEBD1}}(dt) + \mathcal{O}(dt^2)$ which means each time step contributes an error of order dt^2. Calculating the error over the longer time T, since we have a total of $N = T/dt$ steps, the total error is $\frac{T}{dt}\mathcal{O}(dt^2) = \mathcal{O}(dt)$ - a first order error in dt over the course of the entire evolution.

Higher-order Trotter formulae can be generated by considering higher-order approximations of the exponentiated commutator. Relevant to the future discussion is the second-order Trotterisation given by

$$U_{\text{TEBD2}}(dt) = e^{-iH_{\text{even}}\frac{dt}{2}}e^{-iH_{\text{odd}}dt}e^{-iH_{\text{even}}\frac{dt}{2}}, \tag{3.1.5}$$

which has a third order error in the approximation of $U(dt)$ and hence a second order error in the approximation of the overall evolution.

To access the full time evolution up to time T classically, TEBD represents $U_{TEBD}(dt)$ as an matrix product operator (MPO) that gets applied N times to the initial state. The tensor network diagram for a MPS state evolved using MPOs representing the first order Trotterisation is shown in Figure 3.1.

To get the MPO representing the evolution, one can apply sequential SVDs to find the local tensors. Each pair of sites where there is an interaction $e^{-i\hat{h}_{i,i+1}dt}$ has an MPO bond dimension of at most σ^2, whereas the identity interactions have a bond dimension that is trivially 1 [89]. Therefore each application of $e^{-iH_{even}dt}e^{-iH_{odd}dt}$ necessarily has a constant bond dimension σ^2 throughout the chain. Higher order Trotterisation can evolve for longer dt with lower error but may require a higher bond dimension for a single time step.

Repeated application of the time evolution MPOs as required by TEBD can result in an exponential growth in bond dimension for long

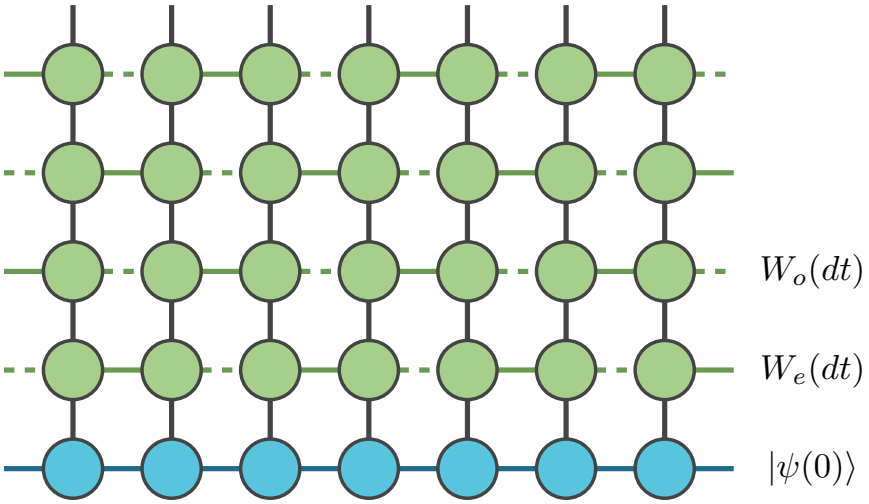


Figure 3.1: **Outline of classical TEBD for a 2 site Hamiltonian:** The initial state $|\psi(0)\rangle$ represented by the blue MPS is evolved by the first order Trotterised TEBD operator which is given by the MPOs on the even and odd site $W_e(dt) = e^{-iH_e dt}$ and $W_o(dt) = e^{-iH_o dt}$ respectively. The dashed virtual indices of the MPOs indicate that the bonds where no local Hamiltonian is applied. Hence, they are trivial with a bond dimension of 1. This diagram shows two applications of $U_{\text{TEBD1}}(dt)$ hence resulting in a total evolution time $T = 2dt$.

times. Therefore, repeated applications of the Trottersised unitary are interleaved with truncations using SVDs to keep the bond dimension under control. However, this is a leading source of error in TEBD. In the context of performing a quantum analogue of MPS time evolution, this SVD truncation step also cannot be directly translated onto a near-term quantum device.

Time dependent variational principle (TDVP)

In comparison to TEBD, which necessarily leaves the MPS manifold, TDVP constructs a set of equations that evolves the state whilst remaining in the MPS manifold. We can use insights from TDVP to stay on the iMPS circuit manifold during the quantum time evolution algorithm.

To understand the key aspects of TDVP for this work, we focus on the application of TDVP to the uniform matrix product state (uMPS) manifold [80]. To begin, consider the time-dependent Schrödinger equation applied to a uMPS with state tensor A,

$$i\frac{\partial}{\partial t}|\psi(A)\rangle = H|\psi(A)\rangle\,. \tag{3.1.6}$$

Generally, this equation's path leaves the MPS manifold immediately. However if we want to restrict the path such that it remains on the manifold - $|\psi(A(t))\rangle$ - then we need to project equation 3.1.6 onto the tangent space of $A(t)$

$$\begin{aligned} i\frac{\partial}{\partial t}|\psi(A(t))\rangle &= P_{A(t)}H|\psi(A(t))\rangle \\ &= \left|\phi(\dot{A};A)\right\rangle \end{aligned} \tag{3.1.7}$$

Here $P_{A(t)}$ is the projector onto the tangent space and $|\phi(B;A)\rangle$ is a representation of the tangent vector at A parameterised by B. Note that finding $\dot{A}$ the tangent vector that provides the best approximation to $H|\psi(A)\rangle$ can be done directly by constructing the projector onto the tangent space or by performing the following minimisation

$$\dot{A} = \arg\min_{B} ||H|\psi(A)\rangle - |\psi(B;A)\rangle||_2^2. \tag{3.1.8}$$

For a formal construction of the projector and subsequent TDVP equations, see Refs [80, 89]. For the quantum time evolution algorithm outlined in this chapter, it is important to highlight that projecting back onto the manifold can be solved using a minimisation problem that can be variationally optimised.

3.1.2 Quantum iMPS time evolution algorithm

We utilise Trottersiation and projecting back onto the iMPS ansatz to perform time evolution in the thermodynamic limit on near-term quantum devices. To evolve the state, like in TEBD, we discretise the time T into time steps of size dt. At each time step, we prepare the state at time t using a parameterised iMPS circuit $|\psi(U(t))\rangle$ with state unitaries $U(t)$. Then we apply a time evolution operator to prepare the state $\left|\psi(t+dt) = e^{-iHdt}|\psi_U(t)\rangle\right\rangle$ - the evolution operator is approximated by a Trottersiation. We subsequently project back onto the iMPS manifold by finding the unitary $U(t+dt)$ that satisfies

$$U(t+dt) = \arg\max_{W} |\langle\psi(W)|\, e^{-iHdt}|\psi(U(t))\rangle\,|. \tag{3.1.9}$$

The projection of the evolved state back onto the iMPS manifold is reminiscent of TDVP. Indeed equation 3.1.9 has a built-in minimisation problem with a fidelity cost function that can be optimised to find $U(t+dt)$. Figure 3.2 summarises the overall time evolution algorithm. Hence, we can use the variational quantum algorithm paradigm to per-

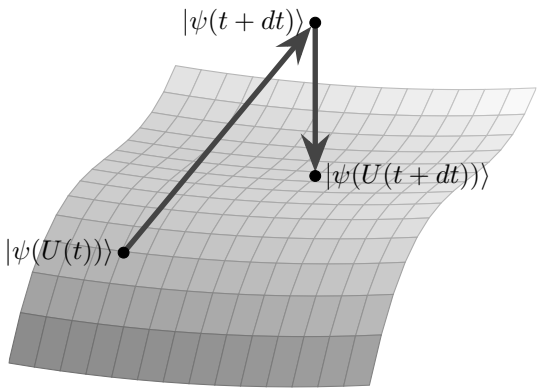
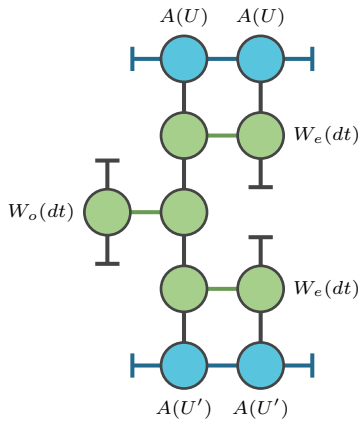


Figure 3.2: **Overview of the time evolution algorithm:** At each time step t the state is represented by $|U(t)\rangle$ on the iMPS manifold. The state is then evolved using a Trotterised evolution operator, which takes it out of the manifold to state $|\psi(t+dt)\rangle$. Subsequently, we project the state back onto the manifold by solving a minimisation problem resulting in the state on the iMPS manifold at time $t+dt$ given by $|U(t+dt)\rangle$.

form the update at each time step.

Due to working in the thermodynamic limit, the fidelity of states is only non-zero when the states are identical. Hence, the cost function has to be modified to calculate the fidelity density between the updated state $|\psi(t+dt)\rangle$ and the parameterised state on the iMPS manifold $|\psi(U')\rangle$. The fidelity density is the leading order eigenvalue of the mixed transfer matrix $E_{U',U}$ between these two states and is given by

$E_{U',U}(dt) =$ 

. (3.1.10)

As in the groundstate optimisation problem, we can construct finite circuits to represent the fidelity density cost function. Therefore, at each step in the time evolution, we perform a variational optimisation on the quantum device to find the updated states at the next step. The details of constructing the fidelity density cost function circuits will be outlined in section 3.3.1. However, first, I will outline the dynamical

model we are interested in investigating using this algorithm.

3.2 Model choice - DQPT

A good test for the proposed quantum time evolution algorithm would be a dynamical model with interesting features in the short to medium time scales. A suitable model, with significant recent analytical and numerical developments, is the dynamical quantum phase transition (DQPT). This model refers to non-analytic behaviour in physical quantities of interest as a function of time [90]. A control parameter usually drives phase transitions; typical examples of control parameters are temperature and pressure. This control parameter is time in the case of DQPT.

A common model used to produce DQPT involves a global quench across an equilibrium critical point. To perform a quench we prepare the groundstate $|\psi_0\rangle$ of an initial Hamiltonian $H_0 = H(\lambda_0)$. $H(\lambda)$ is some general Hamiltonians parameterised by λ. Following this at time $t = 0$, we turn on some real-time evolution under a new Hamiltonian $H_1 = H(\lambda_1)$. The state at time t is therefore represented by $|\psi(t)\rangle = e^{-iH_1 t}|\psi_0\rangle$. To produce DQPT usually involves a global quench across an equilibrium phase transition, including in the case of the 1D TFIM, our main focus in this chapter.

The witness for the DQPT during a global quench is the Loschmidt echo. The Loschmidt echo $\mathcal{L}(t)$ is the fidelity between the evolved state at time t and the initial state and is given by

$$\mathcal{L}(t) = |\langle\psi_0| e^{-iH_1 t} |\psi_0\rangle|^2. \tag{3.2.1}$$

In general, this quantity is exponentially suppressed by the size of the system N. As we are working in the thermodynamic limit, we can define a rate function $\lambda(t)$ that is well-defined as the system size scales and follows

$$\mathcal{L}(t) = e^{-N\lambda(t)}. \tag{3.2.2}$$

In systems that display DQPT, there is non-analytic behaviour in the Loschmidt echo that is present as kinks in the rate function $\lambda(t)$ at certain critical times t_c. These changes are analogous to the non-analytic behaviour of free energy in the context of equilibrium phase transitions. Non-analytic behaviour in the rate function during a global quench has been demonstrated in a number of 1D and 2D system including Chern insulators [91] and the 2D Ising model [92].

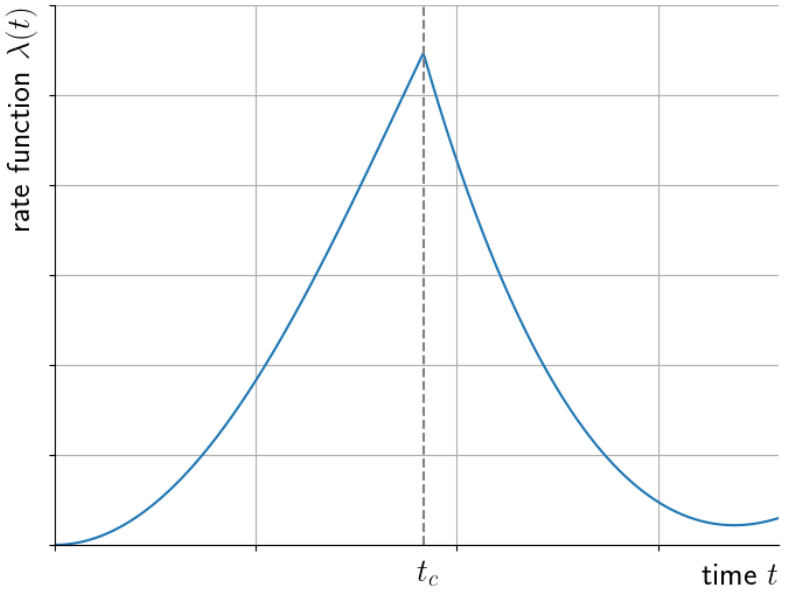


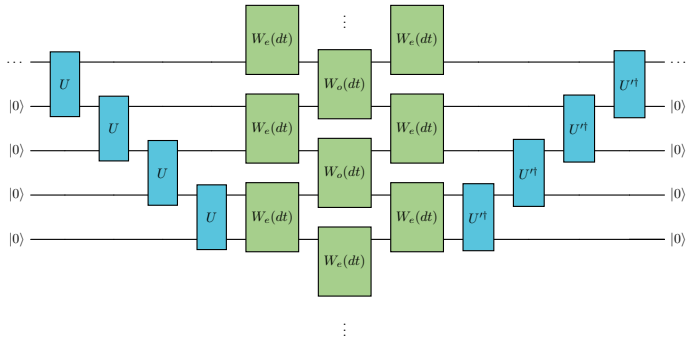
Figure 3.3: **Example dynamical quantum phase transition:** An example plot of rate function over time for a system showing a DQPT. In particular, this is a sketch of the behaviour of the rate function for a global quench of the TFIM across a groundstate critical point. This figure is prepared from real data and comprises of a quench from $g = 1.2$ to $g = 0.5$. Note that there is non-analytic behaviour in the rate function $\lambda(t)$ at some critical time t_c; this is a marker for the presence of the phase transition.

The DQPT demonstrated by the transverse field Ising model is particularly interesting for our discussion. This model is helpful as the analytic behaviour of the rate function is known [93], and this phase transition can be captured well by low bond dimension MPS simulation. Figure 3.3 for a demonstration of these kinks in the rate function for a DQPT using the transverse field Ising model. Therefore, the problem provides a sound test system to demonstrate a proof of concept for our time evolution algorithm.

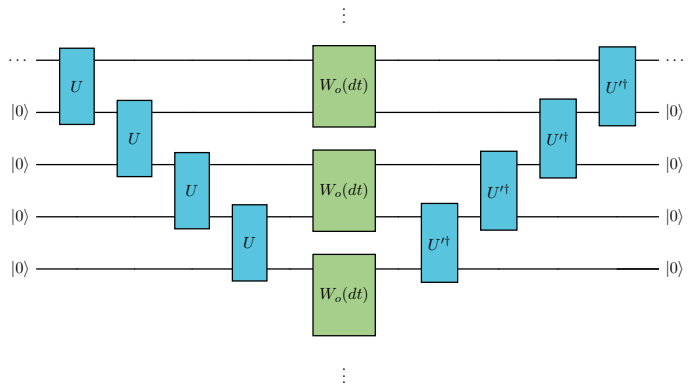
3.3 Adapting for NISQ Devices

3.3.1 Fidelity density cost function

As mentioned in Section 3.1.2, the primary subroutine for implementing the iMPS time evolution algorithm is optimising the fidelity cost function. As the time evolution operator is applied globally, the full circuit for the cost function involves the Trottersied evolution operators



(a)



(b)

Figure 3.4: **Fidelity cost function circuit using the iMPS circuit ansatz:** a) The fidelity cost function with a second order Trotterisation. b) The fidelity cost function with an effective error of $\mathcal{O}(dt)$ according to the TDVP equations due to translational invariance, equivalent to a first-order Trotterisation.

sandwiched between the iMPS circuit representing the state at time t - $|\psi(U(t))\rangle$ - and the ansatz iMPS state to optimise over - $|\psi(U')\rangle$. Figure 3.4a shows the circuit representation of the infinite cost function using a second-order Trotterisation. Note that all discussion in this chapter uses bond dimension 2 iMPS circuits.

We can choose to optimise the fidelity density, thereby compressing this infinite circuit onto a finite one. In addition to compressing to a finite circuit, direct application of this fidelity cost function on near-term quantum devices is challenging due to device performance. Hence, we find adaptations to the cost function by experimenting with current-generation hardware. The following sections outline critical progress in defining the fidelity density cost function using Google's superconducting Sycamore architecture as a test bed for this development.

Order of Trotterisation

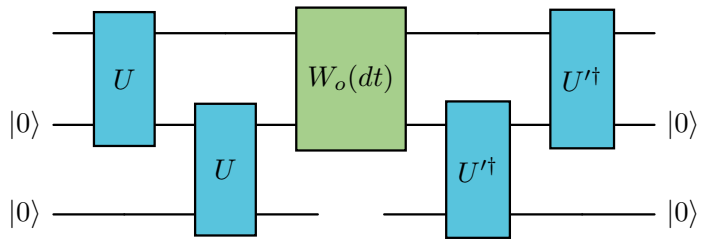
Firstly, implementing the time evolution operator can often be one of the deepest parts of the circuit. The tradeoff between increasing the order of Trotterisation and increasing the circuit depth needs to be carefully considered. With a higher order of Trotterisation the depth of the circuit increases however a larger time step can be accurately simulated as the error per time step is smaller. Therefore, fewer Trotter steps are required for a given evolution. The entire cost function is shown in Figure 3.4a for the second order Trotterisation will perform well for a reasonably large dt. In practicality, circuits like the one shown in Figure 3.4b, showing what looks like a half Trotter step, work much better on devices like Sycamore that are dominated by decoherence errors.

In fact, the circuit shown in Figure 3.4b corresponds to an error equivalent to a first-order Trotterisation. The reduced size of the circuit is possible since the system we are working with is translationally invariant. Hence projecting back onto the iMPS manifold using the TDVP equations suggest the circuit shown in Figure 3.4b produces errors in dt of order $\mathcal{O}(dt)$. Hence, this results in a higher order error than one would expect naively when treating Trotter errors.

Power method to calculate fidelity density

As we are working in the thermodynamic limit, we are interested in calculating the fidelity density λ. In the circuit picture, this can be found by calculating the leading order eigenvalue of the mixed transfer matrix $E_{U',U}$ given by

$E_{U,U'}(dt) =$



(3.3.1)

Similarly to groundstate optimisation, we can calculate the fidelity density using environment unitaries. Due to this being a mixed transfer matrix, the left environment is no longer the identity. Therefore, we need to calculate the left and right environment unitaries L and R, respectively. Figure 3.5a shows the overall circuit for the fidelity density.

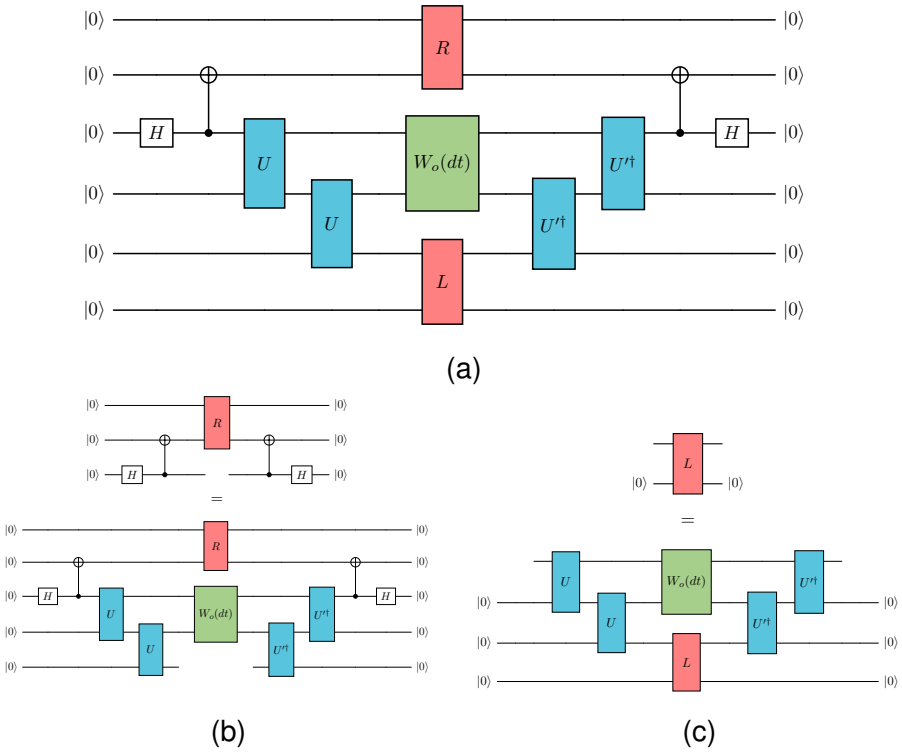


Figure 3.5: **Exact circuits for calculating the fidelity density for the time evolution cost function.** Figure 3.5a shows the full fidelity density cost function circuit. Figures 3.5b and 3.5c show the fixed point equations that need to be satisfied to represent the environments.

If we are to explicitly calculate the environment unitaries L and R, we now have to optimise two fixed point equations simultaneously. The circuit diagrams for these equations are given in figure 3.5b and 3.5c. These fixed point equations vary each time the ansatz unitary U' is modified - meaning the optimiser changes the unitaries each time. The combined cost function to solve these fixed points and optimise the fidelity density becomes prohibitively expensive for time evolution.

However, we do not need to explicitly solve fixed point equations to calculate the fidelity density λ. Instead, we utilise the power method to approximate λ. The power method states that given an approximate representation of the left and right eigenvectors $\widetilde{L}$ and $\widetilde{R}$ of the mixed transfer matrix $E_{U,U'}$, the leading order eigenvalue of the mixed transfer matrix, and therefore the fidelity density, can be approximated by

$$\lambda = \lim_{n\to\infty} \frac{\widetilde{L} E^n_{U,U'} \widetilde{R}}{\widetilde{L} E^{n-1}_{U,U'} \widetilde{R}} = \lim_{n\to\infty} \frac{C_n(U,U')}{C_{n-1}(U,U')}. \tag{3.3.2}$$

It turns out that, given an excellent approximation to the eigenvectors, even a low order n for the power method provides a route to estimate the fidelity density. In addition, the fraction need not be calculated as

n applications of the transfer matrix approximate the nth order power on the fidelity density λ, i.e. $C_n(U, U') \approx \lambda^n$.

Optimising States

We test the power method for state optimisation to demonstrate low orders of the power method can accurately calculate fidelity density. In this experiment, we randomly initialise an arbitrary iMPS circuit with a state unitary U which is known. We then optimise the fidelity density cost function (without time evolution) to relearn U from a random initialisation of the state ansatz U'. This method amounts to optimising the overlap between the known state $|\psi(U)\rangle$ and the parameterised state $|\psi(U')\rangle$. An example circuit for $C_n(U, U')$ from the power method is shown in Figure 3.6. Here we show an order $n = 6$ of the power method with the approximations $L = \mathbb{I}$ and $R = |0\rangle\langle 0|$.

There is a tradeoff between the circuit's depth and the power method's accuracy. Fortunately, this method of calculating overlaps has a natural error mitigation method, which involves rescaling the calculations of $C_n(U, U')$ by the Loschmidt echo circuit $C_n(U, U)$. The Loschmidt echo circuit should always resolve to the identity, but depolarisation errors increase as the depth of the circuit increases. Hence, the Loschmidt echo provides a good way of rescaling the output of the original power method circuits with the same complexity to account for depolarisation errors.

Before performing state optimisation, we tested various orders of the power method on Sycamore to determine the most accurate order given the decoherence errors. Figure 3.6b shows the performance of this technique when calculating fidelity density between two arbitrary states for various orders of the power method. We see that by $n = 4$, there is good agreement between the exact fidelity density λ and the one calculated on the device. The point at $n = 6$ seems to be an outlier due to errors in calculating the Loschmidt echo rescaling. Through interpolation, we find that an estimate of the $n = 6$ point without this error lies much closer to the accurate λ. As we can see by the variance shown in the figure, for order $n > 7$ of the power method, there is a significant error in calculating fidelity density caused by decoherence errors on the device that the Loschmidt rescaling can no longer account for.

We then perform state optimisation using order $n = 4$ of the power method. Figure 3.6d shows the outcomes of this optimisation experiment. For this optimisation, this technique works well, with a fidelity of

0.98 within 50 steps of SPSA optimisation. Further optimisation steps do not improve the performance due to errors on the device. These results are promising, but due to the added depth of the time evolution operators, we need to carefully construct the approximate eigenvectors of the mixed transfer matrix and the order of the power method before implementing this algorithm to perform time evolution.

Approximating the eigenvectors

A central challenge in utilising the power method to calculate fidelity density is finding good approximations to the left and right eigenvectors. If the left and right eigenvectors are exact, the power method is equivalent to our explicit calculation for fidelity density at $n = 1$. In the profiling cost function section below, we perform some preliminary testing on Sycamore of approximations to the environment for $n = 2$ where we use a classical simulator of the power method cost functions to determine if they can effectively reproduce the dynamical phase transition of the transverse field Ising model.

We find the left and right environments are well approximated to the identity, and a single additional iteration of the state unitary $U(t)$ contracted with its conjugate $U^\dagger(t)$ respectively. Heuristically, this makes sense as the Trotterised evolution operator will act with a relatively small dt, and these approximate left and right environments should be accurate to order $\mathcal{O}(dt^2)$.

Reduced parameterisation

During groundstate optimisation using iMPS circuits, we found that using reduced parameterisations of arbitrary two-qubit unitaries was necessary to make the optimisation feasible. Inspired by the groundstate ansatz, we use a modified repeated ansatz shown in Figure 3.7a for time evolution. This ansatz reduces the number of parameters needed from a full $SU(4)$ unitary, requiring 15 parameters, down to 8 parameters.

In addition, as the time evolution unitaries act over two qubits, we can readily parameterise them using a KaK decomposition [76]. Applying the KaK decomposition and optimising the compilation of the time evolution unitaries leads to circuits like the one shown in Figure 3.7b.

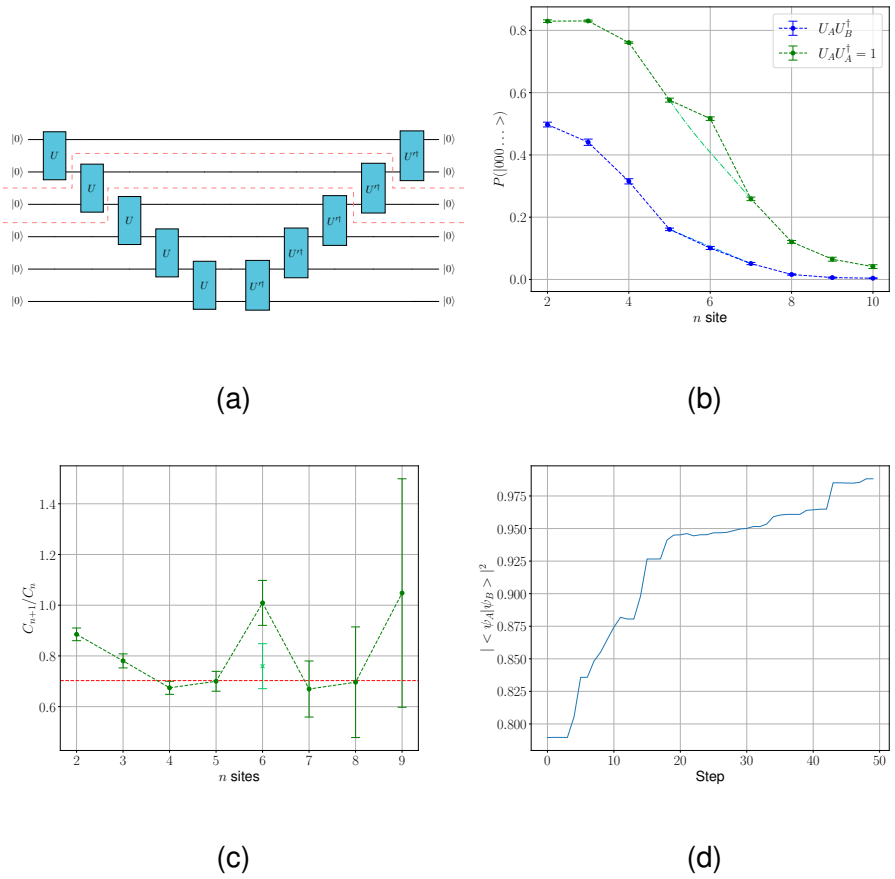
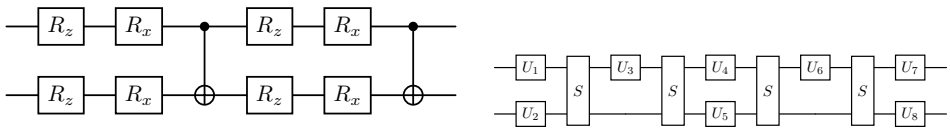


Figure 3.6: **State optimisation results on Sycamore:** a) The circuit representing the $C_n(U, U')$ term required to calculate the fidelity density using the power method. This circuit represents $n = 5$ and works by iteratively applying n copies of the transfer matrix highlighted in red and calculating the probability of all zero state $|0\rangle^{\otimes(n+1)}$. b) The raw output of measuring $C_n(U_A, U_B)$ for various values of n is given by the blue curve. Depolarisation error is a significant contribution to the decay of the signal. This can be corrected for by measuring the Loschmidt echo $C_n(U_A, U_A)$ at each value of n and rescaling. The error bars give the standard deviation from 10 repetitions of the measurement. We use 10^6 samples so sampling error is likely limited and this is predominantly influenced by noise. c) The estimates for fidelity density using the power method at each value of n. These results include the Loschmidt rescaling to mitigate for errors. The dashed line is the exact value calculated classically. By $n = 4$, the measured value of the fidelity density on the device closely matches the exact value. As n increases, the results seem to converge to the correct value. As $n > 7$, the deplorising error dominates the results. Note that the point at $n = 6$ is an outlier, likely due to an error in calculating the Loschmidt rescaling. By interpolating the Loschmidt rescaling as shown in b), this value can be corrected for. d) Stochastic optimisation of the fidelity density circuits for $n = 4$ to show that these fidelity density circuits can be used for state preparation.



(a) (b)

Figure 3.7: **Parameterisation of unitaries for time evolution.** Figure a) shows the parameterisation for the state unitaries. The gates R_x and R_y are rotation gates given by $R_{\sigma_i} = \exp(-i\sigma_i\theta)$ where σ_i is the respective Pauli operator. This decomposition is a reduced parameterisation of the full $SU(4)$ unitary, which requires 15 parameters down to a parameterisation which only requires eight parameters, one for each rotation gate. Figure b) shows the parameterisation of the time evolution unitaries after being decomposed into a Kak decomposition and compiled by Google's compiler for the Sycamore processor. The gates labelled U_i represent arbitrary rotation gates and S represents the $\sqrt{iSWAP}$ two-qubit gate, which is native to Sycamore.

Profiling Cost Function on Sycamore

We profile the cost function at various time steps on Sycamore to verify the fidelity density cost function circuits. To do so, we select a set of time steps and calculate the optimal parameters at the start and the end of the optimisation. We draw a linear path from the initial point in the optimisation to the optimal point and beyond, selecting evenly spaced points on this path. The fidelity cost function circuits are run on Sycamore at each point to get a picture of the cost function landscape on the device.

Several error mitigation techniques were used to give the best possible performance on the device. We use a method we refer to as Loschmidt rescaling to account for depolarisation error (note that this is not directly related to the Loschmidt echo in DQPT). In this case, we require circuits with equivalent complexity to the Trotterised evolution operator that resolves to the identity. Analogous to the rescaling used in groundstate optimisation, we are able to prepare a circuit with a similar complexity to the target circuit with a known output value (an all $|0\rangle$ state). A simple way to implement this is to perform the Trotterised evolution with an extremely small dt using the initial state unitary and its Hermitian conjugate as the state ansatz on either side of the operator. From 3.6 this technique has a limitation that when the circuits become too large the signal from the circuit is too small to accurately rescale. In addition to Loschmidt rescaling, we used qubit selection and qubit averaging to account for stochastic variations in device performance over the run of the experiment. Figure 3.8b shows the cost function

profiling experiment results for the chosen orders of the power method and optimal environment. The key takeaway from this experiment is that the optimal parameters, given at parameter step 5, are the optimal points of the cost function both when calculated classically and on Sycamore's processor.

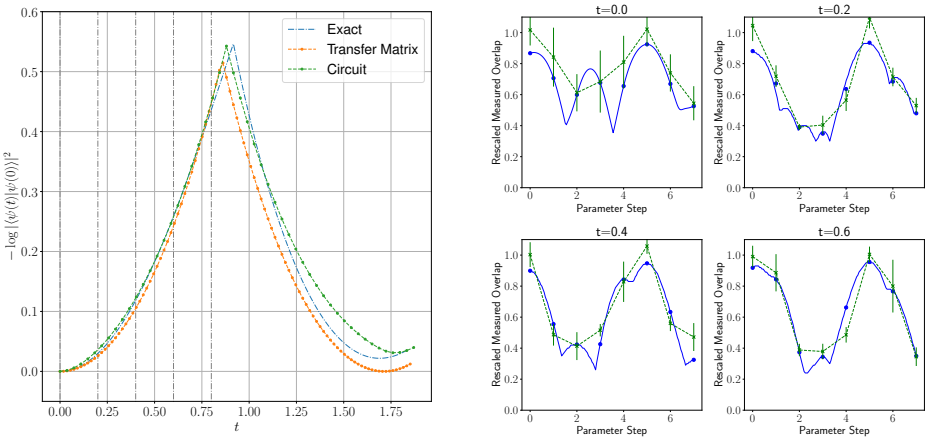
The results from Sycamore suggest this cost function can perform time evolution using iMPS circuits on near-term devices. Figure 3.9 outlines the approximate fidelity density cost functions in both the space-like and time-like layout. However, running the entire time evolution requires careful consideration of how to effectively perform the optimisation, including accounting for noise-resilient and shot-efficient routines.

3.3.2 Tuning Optimisation

Due to noise on NISQ devices, the choice and tuning of the classical optimisation algorithm are vital in the success of any quantum variational algorithm [94]. Many native quantum optimisation algorithms work by utilising the properties of the underlying parameterised quantum circuit to perform classical optimisation. However, we choose to use SPSA for our optimisation due to its noise resilience and the fact that it has been used in other variational experiments using near-term quantum devices.

Despite this choice, sampling costs can be a fundamental limitation for variational quantum algorithms, potentially preventing quantum advantage [95].The issue of sampling can be particularly problematic for platforms like the trapped-ion systems, where gate times can be of order $10^2 \mu s$ [96].

At first glance, our time evolution algorithm is in a dire situation, requiring an optimisation loop at each time step. However, choosing a good initialisation can significantly reduce the number of samples an optimiser needs. When time evolving a state, we expect the parameters in parameter space to vary smoothly. Therefore, given information about the parameters at previous time steps, the optimisation can be initialised by a simple extrapolation that can be performed efficiently classically. The extrapolation method can vary in complexity, from something as simple as polynomial spline to as complicated as a classical machine learning model. The complexity of the classical extrapolation method needed will be correlated with the shape of the paths in parameters space. A good choice in circuit parameterisation could lead to a more straightforward path and hence a less compli-



(a) (b)

Figure 3.8: **Profiling fidelity density cost function on Sycamore:** Figure a) shows classically simulated data to measure the dynamical quantum phase transition of the transverse field Ising model through a global quench from $g = 1.2$ to $g = 0.5$. The blue curve shows the analytically exact result from the literature. The orange dashed curve the evolution produced by optimising the leading order eigenvalue of the mixed transfer matrix classically. This curve uses the same reduced parameterisation for the state unitaries that is used on the circuit. The dashed-dotted green curve shows the optimisation of noiseless simulation of the proposed fidelity density cost function circuits. This shows that the circuit cost function can faithfully track the behaviour of the dynamical phase transition. Figure b) shows the cost function evaluated on Rainbow. The parameters at each time slice highlighted by the grey vertical bars in a) are taken, and a line is drawn from the initial parameter at time t and the optimal parameters at time $t + dt$. Equally, space points are taken along this line and the value of the circuit cost function is measured on Sycamore. Parameter step 0 is the parameter at the initial point of the optimisation, and step 5 is the optimal point. The green curves show the measured values on Sycamore and the blue curve shows the exact. The circuit values are error-mitigated using Loschmidt rescaling. Note that the optimal value calculated classically matches the optimal parameters on the device.

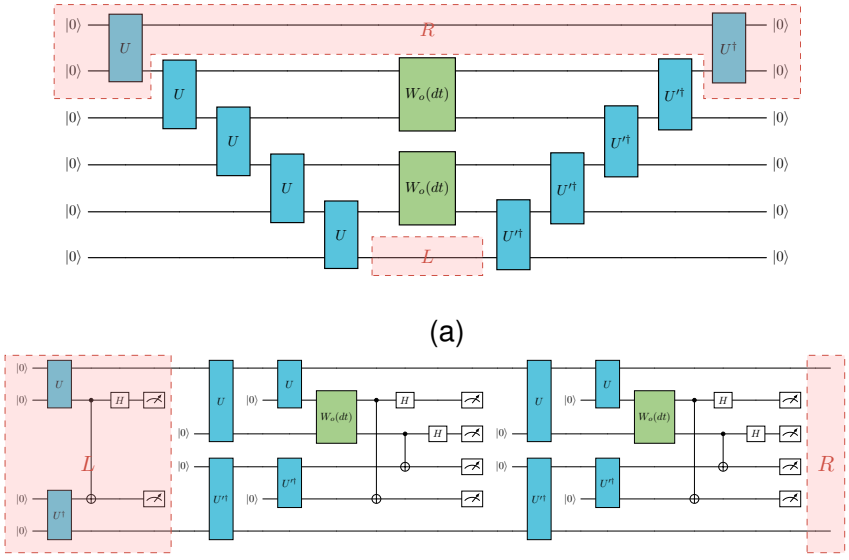


Figure 3.9: **Approximate cost function circuits for time evolution:** Both circuits show second-order power method circuits with a time evolution operator (shown in green) with errors equivalent to the first-order Trotterisation of the evolution operator. The state unitaries are given in blue, and the state unitary at time t is given by U, whereas the ansatz state unitary to optimise over is given by W. The regions highlighted in red represent the environments. Figure a) shows a space-like layout of the circuit, which is in left canonical form. The left environment L is the identity and the right environment R is repeated applications of the state unitary U. Figure b) shows a time-like layout of the circuit in the right canonical form. Hence, the left environment L is repeated applications of the state unitary U and the right environment is the identity, amounting to not measuring the right environment qubits.

cated extrapolator. When viewed in this paradigm, the quantum computer can be regarded as making stochastic corrections on a classical model for the evolution of the state.

3.3.3 Dynamics on a trapped-ion device

Experiment outline

We performed an entire run of the time evolution algorithm on Quantinuum's trapped-ion H1 device. We initialise the evolution with the ground state of the TFIM for $g = 1.5$. We perform a quench using the time evolution algorithm with a Hamiltonian of the TFIM at $g = 0.2$. The Loschmidt echo for the time-evolved state is measured at each time step to determine whether a DQPT is observed.

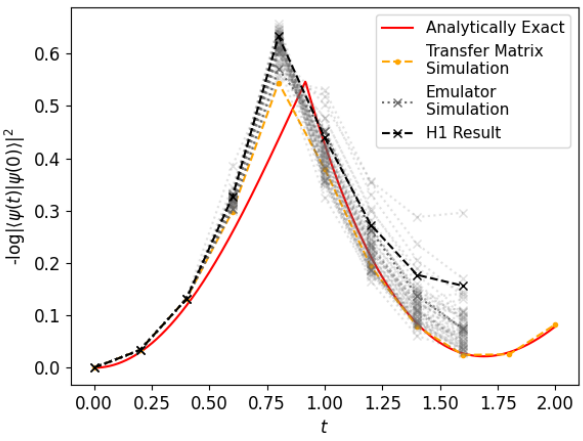
We use the time-like versions of the fidelity density circuits to evaluate the cost function. The circuit shown in Figure 3.9b represents $C_2(A, B)$ from the power method (Equation 3.3.2) and is the circuit we ran on the device. We use the same approximations to the environment used in the Sycamore circuits with a $SWAP$ gate added to the ansatz to map the space-like ansatz used on Sycamore to the time-like form. Whereas the space-like circuit was well suited to Sycamore, this time-like rewrite uses the longer coherence times and presence of mid-circuit measurement of the qubits on H1. Given the excellent coherence times on Quantinuum's device, we found that error mitigation schemes such as Loschmidt rescaling are unnecessary. However, the restricted shot budget placed significant limitations on performing variational quantum algorithms on this device.

We optimise the cost function using SPSA with an initialisation based on a linear extrapolation. This simple classical method worked due to a fortunate choice in the parameterisation for state unitaries U. Due to this choice, there is a restriction in the gauge of the state unitaries so that the parameters vary relatively linearly in parameter space during the evolution. Using this initialisation scheme, we can reduce the shots required for a full time evolution by around three orders of magnitude, effectively making our algorithm feasible on near-term quantum devices. To utilise this extrapolation in the experiment, we classically find the first two steps in the optimisation by optimising the classical transfer matrix representation of the fidelity cost function. We then use these initial parameters to seed the next steps' extrapolation.

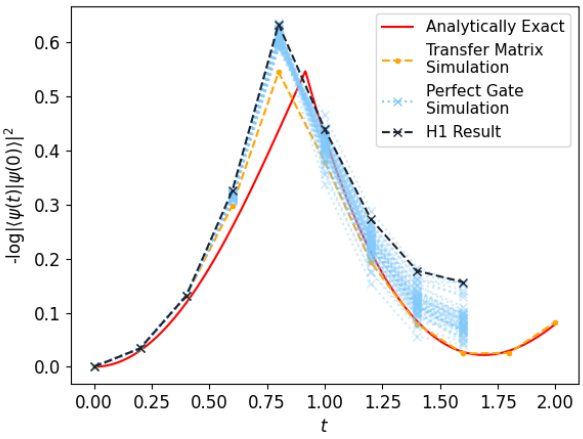
Results

Figure 3.10 shows the results of using our time evolution algorithm to investigate a DQPT of the transverse field Ising model. This figure shows the results of this experiment for a single run on H1 overlaid (shown in black) over 50 runs on the classical emulator of H1 (shown in grey).

The results from the H1 device and emulator qualitatively show the DQPT clearly. A deviation in time is seen both in the results and using the classical in ansatz simulation using the transfer matrix. Therefore, the deviation is likely due to the reduced parameterisation of the state unitaries. The variance of the 50 emulator runs could be due to either fidelity error due to device performance or sampling error, as we were limited to 6000 shots per optimisation. In an ideal scenario a higher shot budget would be possible for improved optimisation, up until the device errors begin to dominate. Figure 3.10b shows a further simulation with the same number of shots and gates with no noise. The noiseless simulation's variance is similar to the emulator's (which accounts for the gate noise present on H1). This suggests that this experiment was limited by sampling error instead of device fidelity. As near-term devices develop, shot budgets will likely increase significantly, suggesting that this algorithm will improve considerably with device performance. In addition, more sophisticated initialisation schemes could be investigated which would utilise the currently available shot budget better.



(a)



(b)

Figure 3.10: **Full run of time evolution on Quantinuum's H1 device:** Figure a) shows the results of running the full time evolution algorithm using the time-like fidelity density cost function on Quantinuum's H1 device. The results show the recurrence behaviour we expect from the dynamical quantum phase transition of the transverse field Ising model. As previously, the red curve and the yellow curve show the analytically exact and transfer matrix simulation respectively. The grey curves show 50 runs on the emulator for the H1 device, giving an idea of the variance of this method. The black curve shows a run on the H1 processor. A significantly optimised SPSA was used with only approximately 6000 shots per time step. We utilise classical linear extrapolations of previous time step parameters to initialise the optimisation. Figure b) demonstrates the variance expected with perfect gates using the same number of shots. As can be seen the variance is still large, suggesting the sources of the variance are predominantly due to sampling noise as opposed to device fidelity.

3.4 Discussion

In this chapter, we develop a proof of concept for a variational algorithm to time-evolve quantum states using the quantum iMPS ansatz. This work involved the implementation of a fidelity density cost function that evolves the state at each time step using a Trotter evolution and subsequently projects the state back down onto the quantum iMPS manifold. We demonstrate the feasibility of this algorithm for probing a subtle quantum dynamical problem, the dynamical quantum phase transition of the transverse field Ising model, using superconducting and trapped-ion hardware. To do so required careful consideration of each type of architecture and an algorithm readily adaptable to each device's individual strengths. In addition, an initialisation strategy is proposed, significantly reducing the sampling requirements for the optimisation steps in the time evolution after some initial time steps. Consequently, the results suggest that current-generation quantum computers can capture interesting features of quantum dynamical systems.

Though this proof of concept was successful, work is still required to get true quantum advantage when dynamically simulating quantum systems. As with the groundstate optimisation problem, a clear route to this would be to increase the bond dimension. This is particularly relevant during time evolution as we expect the entanglement of states to grow systematically with each Trotter step. Hence, a straightforward adaptation of this algorithm, where the bond dimension of the ansatz state in the fidelity density cost function is increased, could provide a route to time-evolving states beyond what is currently feasible using standard classical MPS methods.

A core open question remains how to efficiently increase bond dimension whilst maintaining reasonable sampling. Initialising the quantum optimisation using a simple classical model proved to be a powerful method for reducing the sampling complexity in this instance. This procedure would have to be generalised to incorporate increasing the bond dimension. As the bond dimension increases, the number of parameters required to represent a state unitary increases significantly. Therefore, more sophisticated classical models, such as polynomial splines or machine learning, may be necessary to account for the increase in the size of the parameter domain.

During the experiment with the H1 device, we noticed that the device's fidelity was not the dominant source of error. Therefore, the tradeoff between the order of Trotterisation and the time step size may not be optimal. This device profile may accommodate a higher order

Trotter step and, thus, an increased time step while still providing good accuracy. Increasing the time step offers additional benefits, such as reducing the amount of sampling necessary to evolve for a given time as the time step sizes are larger. Hence, subsequent experiments could look into how the algorithm could be tuned further for the device characteristics of H1. This feature of the algorithm, to be tunable to particular device characteristics, emphasises the portability of this algorithm to different NISQ devices.

Finally, this time evolution algorithm prioritises state fidelities; however, this may not be a good metric to preserve for achieving long time evolution. Recent developments in classical tensor network methods for time evolution have found that preserving local properties of states, namely local reduced density matrices, is a more efficient way of achieving long time evolution. The underlying principle behind this choice is that relatively local correlations often bound physical properties of interest. Therefore, one can achieve long time evolution by prioritising local observables, which inherently limit the bond dimension growth during evolution. Analogues of this idea using the iMPS ansatz are further explored in the following chapter.

Chapter 4

Local density matrix evolution

A problem of particular interest in condensed matter physics is how macroscopic thermal properties arise from non-equilibrium dynamics [97]. Answering this question through simulating real-time dynamics is particularly challenging for large systems as resource requirements grow significantly with system size [98]. Applying standard tensor network techniques to evolve the entire state, such as in the case of TEBD, for MPSs would result in an exponential increase in the bond dimension with time [89]. Therefore, accurate simulation is only possible for a short time.

Do we need to represent the entire state at all times? Many quantities of interest, such as single spin polarisation, are local [99]. Furthermore, thermodynamic properties are often calculated by breaking down the quantity into a sum of local operators. Perhaps all that is required is preserving the state up to some local patch, i.e. preserving the local density matrix of the state during evolution. Further motivation for this approach can be found given that thermal density matrices, a representation of quantum systems at long times, have efficient tensor network representations. Given that thermal correlations often decay exponentially [100], there is strong evidence that 1D and 2D Gibbs states above a finite β can be represented accurately by finite bond dimension tensor networks. Hence, given a thermal correlation length ϵ, simulating a patch size of order $\mathcal{O}(\epsilon)$ accurately allows for the calculation of any thermal properties once thermalised.

This section outlines an approach to perform real-time evolution whilst optimising the local density matrix at each step. We are in part inspired by several recent approaches in this field with similar ideas, including density matrix truncation (DMT) [101] and dissipation-assisted operator evolution (DAOE) [102]. Previous approaches focus on using MPOs to represent density matrices and evolve the state whilst pre-

serving local density matrices through methods including preserving a hierarchy of operators about the boundaries or artificial dissipation. Our approach performs the evolution on a iMPS state ansatz whilst performing the evolution by preserving local density matrices using a trace distance metric. A benefit of this method is that we utilise the structure of the time evolution algorithm proposed in Chapter 3 with a modified cost function that can be implemented on a quantum device.

This chapter first introduces a classical algorithm for performing the time evolution. We apply this algorithm to the dynamical phase transition problem outlined in Chapter 3. Finally, we propose analogous local density matrix cost functions that can be represented on a quantum device. The work in this chapter is my own and was done in collaboration with Andrew Green.

4.1 Preserving local information

Generally, time evolution algorithms look to preserve the full state fidelity at all times. However, doing so requires significant computational resources. As an example, consider quench dynamics in 1D spin systems. Usually, quantum quenches start with easy-to-prepare states, such as the ground states of gapped Hamiltonians, where an efficient MPS representation is known to exist. As the system evolves, the bond dimension D needed to accurately capture the unitary dynamics of arbitrary Hamiltonian scales as $\mathcal{O}(\exp(t))$ [103]. We can heuristically motivate this result by recognising that the Trotterised propagator e^{-iHdt} in TEBD can be represented by an MPO with a bond dimension χ. The number of MPOs needed to reach a total evolution time t scales linearly with t. Evolving the state involves contracting this sequence of MPOs with the initial MPS. The bond dimension of the resultant MPS when an MPO is applied to it is the product of the original MPS bond dimension and the MPO bond dimension, hence the exponential scaling in the bond dimension when evolving to the total time t [48].

However, on the other side of long-time evolution, thermalised 1D systems are known to have an efficient MPO representation [104]. Thermal density matrices are represented by the Gibbs state $\rho = e^{-\beta H}/Z$, where Z is the partition function, H is the Hamiltonian and β is the inverse temperature. Thermal expectation values of an operator A are calculated by sampling the Gibbs state

$$\langle A \rangle_\beta = \mathrm{Tr}\left[\frac{Ae^{-\beta H}}{Z}\right], \tag{4.1.1}$$

where Z is the partition function. It is known that the Gibbs state shows a decay in correlations between observables separated by a distance l on a lattice. In 1D translationally invariant systems, there is an exponential decay in correlations [105], namely correlations decay as $e^{-l/\zeta}$ where ζ is some thermal correlation length. This results in the bond dimensions required for sampling thermal systems being dependent on intrinsic properties such as the thermal correlation length and independent of the system size [100, 106].

We propose constructing classical algorithms whereby the quantity preserved is not the global fidelity but the observables over some local patch. In 1D, results suggest that the D needed to represent local observables on k sites with error ϵ is $D \leq \exp(k/\epsilon)$. Hence, heuristically, algorithms that can maintain this bond dimension up to some thermalisation time can potentially simulate thermalising dynamics for

local observables. Several recent algorithms have been proposed that prioritise preserving local observables whilst controlling bond dimension. Examples include density matrix truncation, where a hierarchy of operators is used to preserve information close to the boundary in 1D systems [101], the dissipation-assisted operator evolution that uses dissipation operators to remove weights from non-local operators [102], and the time evolution of local information where the dynamics of an information lattice are modelled directly [99]. These algorithms can often simulate thermalising systems with even lower bond dimensions than the theoretical limits suggest. This is likely due to underlying structures in the Hamiltonians we are interested in simulating that results in intrinsic properties requiring fewer resources than the theoretical results.

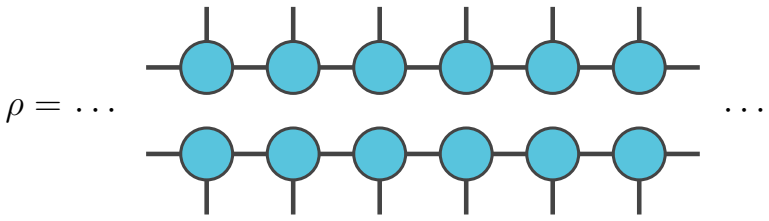
In this chapter, we devise our algorithm that preserves local information during time evolution. We adapt the time evolution algorithm proposed in Chapter 3, where we preserve local density matrices instead of preserving the global fidelity of a state. This adaptation primarily requires rewriting the cost function that projects the state back onto the iMPS manifold. We begin by discussing a classical version of this cost function and propose the adaption to the quantum time evolution algorithm that has the potential to be applied on current NISQ devices.

4.2 Evolving local density matrix

To construct a cost function that preserves local density matrices, we first propose a method to represent local density matrices classically using the iMPS ansatz. Once we have this representation, we can construct a classical cost function analogous to the fidelity density one that preserves a metric based on local density matrices. The requirements for this construction include remaining within the thermodynamic limit and being readily translatable to a quantum cost function. We also propose methods for applying time evolution operators to the density matrix representation.

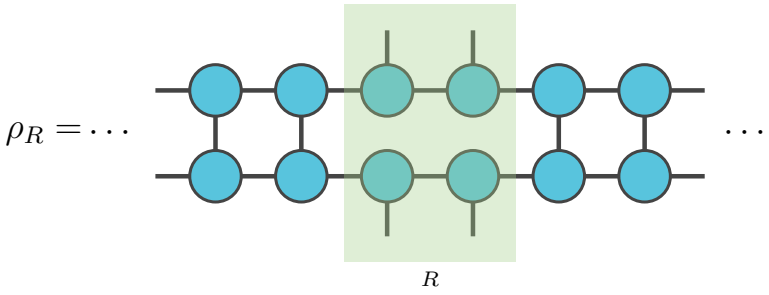
Density matrices are constructed by calculating the outer product of a state and its adjoint. The same operation can be applied to iMPS states to construct the density matrix $\rho = |\psi\rangle \langle\psi|$. In the thermodynamic limit the states $|\psi\rangle$ are represented by iMPS, an infinite chain of

repeated state tensors. Therefore ρ is written diagrammatically as



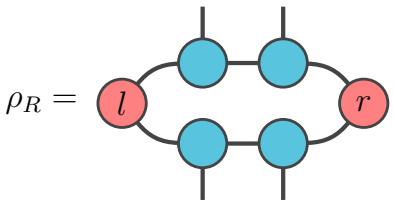
. (4.2.1)

To construct the local density matrix ρ_R over a region R of size L using this ansatz requires taking the partial trace over all the legs in ρ not in the region. Hence, ρ_R can be written as



. (4.2.2)

The chain of traces can be replaced by the action of the environments on either side of the region R. Assuming that r and l represent the right and left eigenvectors of the state transfer matrix, ρ_R can be written as



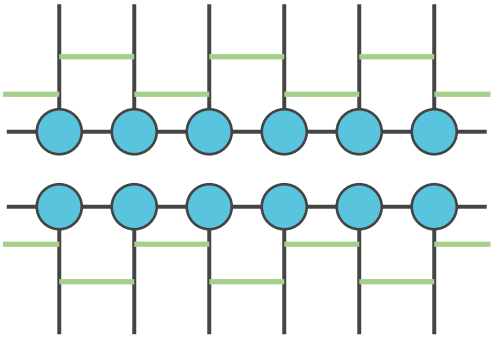
. (4.2.3)

Note that if we are in left canonical form, l will reduce to the identity.

We can extend this prescription for local density matrices to include time evolution operators. Assuming the iMPS state tensor at the initial time is $A(0)$, to time evolve the density matrix at initial time $\rho(A(0))$ up to a time t under a Hamiltonian H requires the application of a propagator $U(t) = e^{-iHt}$ such that

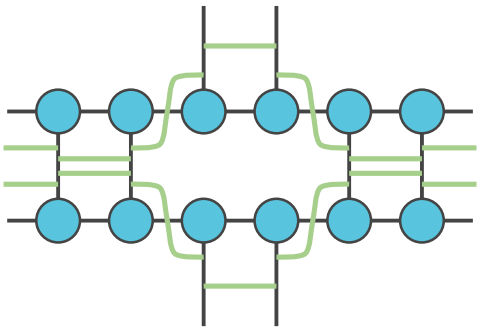
$$\rho(A(0);t) = U(t)\rho(A(0))U^{\dagger}(t). \tag{4.2.4}$$

As with the state evolution algorithm, the propagator $U(t)$ can be constructed using Trotterisation. Consider the Trotterisation of a two-site Hamiltonian up to first order; the evolution of the density matrix can be represented diagrammatically as

$\rho(A(0);t) = \cdots$  $\cdots$

. (4.2.5)

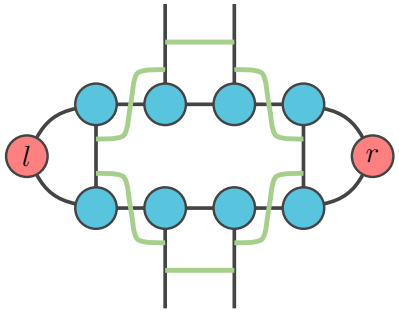
Note that in this diagram the green connections represent the even and odd Trotterised evolution operators. The conjugation of operators in the adjoint space is implied. As in the density matrix with no evolution case, we represent the local density matrix over a region R of size L by performing a partial trace over the infinite chain. Selecting the region R to be of size 2, the evolved local density matrix is

$\rho_R(A(0);t) = \cdots$  $\cdots$

. (4.2.6)

Given the fact that the propagator $U(t)$ is a unitary and therefore satisfies $UU^\dagger = U^\dagger U = I$, this infinite contraction can be reduced using this condition and the environments l and r of the state transfer matrix.

Hence the reduced contraction of $\rho_R(t)$ can be written as

$\rho_R(A(0);t) =$ 

. (4.2.7)

Note that this construction is specific to the order of Trotterisation and region choice. Different orders of Trotterisation or choice of region may change the structure of the diagrams needed to represent local density matrices. Additionally, this local density matrix represents only one application of the Trotter operators. Multiple applications of the Trotter operators will also extend the diagrams.

Now that we can construct local density matrices, the following section outlines constructing cost functions based on preserving the trace distance between the local density matrix of an evolved state with the local density matrix of an ansatz state to be optimised.

4.3 Optimising local cost function

As in Chapter 3, we can formulate the time evolution algorithm within the framework of a variational algorithm. We discretise the overall evolution time T into steps of size dt, and at each step, we encode the evolution within the optimisation of a cost function. We perform this optimisation over a region R of length L. Therefore, note that all density matrices $\rho(\cdot)$ below are constructed over the patch R. The target density matrix is the time-evolved reduced density matrix $\rho(A(t);dt)$ that we construct by applying the evolution propagator $U(dt)$ to the local reduced density matrix $\rho(A(t))$ as outlined in Seciton 4.2. We optimise the distance between $\rho(A(t);dt)$ and the reduced density matrix of the ansatz state $\rho(A')$ within the left canonical iMPS manifold. We can use several different metrics to perform this optimisation; the one used in this chapter is the trace distance of the local reduced density matrices. Hence, the state tensor within the ansatz at the time t is given by

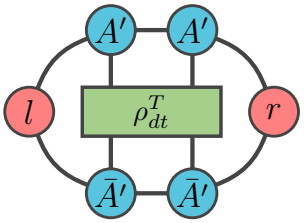
$$A(t) = \arg\min_{W} Tr[(\rho(A(t);dt) - \rho(A'))^2]. \quad (4.3.1)$$

The procedure is identical to the time evolution algorithm outlined in Chapter 3. In fact, as $L \to \infty$, the fidelity density cost function is identical to the reduced density matrix cost function. We expect that there will be some minimal patch size $L = L_{therm}$ that needs to be preserved such that the dynamics and behaviour of the state are preserved faithfully. However, in most instances, we expect L_{therm} to be sufficiently small due to thermal correlations decaying exponentially in 1D. The bond dimension on $\rho(A')$ needed to track the dynamics accurately is also bounded by the size of the patch (L_{therm}) necessary to preserve dynamics.

4.3.1 Derivative of local cost function

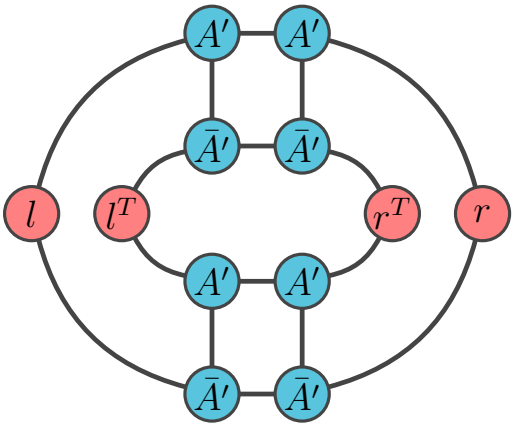
To optimise this cost function using gradient-based methods, this section shows how to calculate the derivative of the local density matrix cost function with respect to $\bar{A}'$. The cost function has two terms which depend on $\bar{A}'$, namely $\mathrm{Tr}\Big[\rho(A')\rho^\dagger(A')\Big]$ and $\mathrm{Tr}\Big[\rho(A(t);dt)\rho^\dagger(A')\Big]$. Diagrammatically, these two terms for a patch region R of size $L = 2$ can be written classically as

$\mathrm{Tr}\,\rho(A(0);dt)\rho^\dagger(A') =$



(4.3.2)

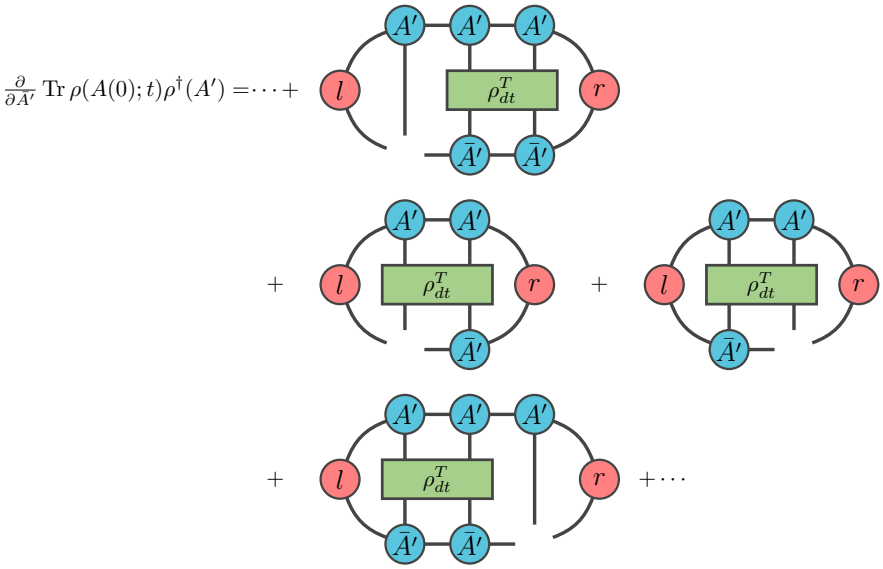
$\mathrm{Tr}\,\rho(A')\rho^\dagger(A') =$



(4.3.3)

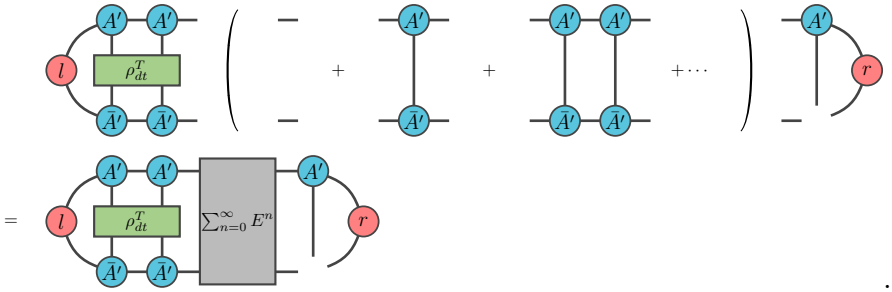
where $\rho_{dt} = \rho(A(t);dt)$. Calculating the derivative requires removing $\bar{A}'$ at each position where it appears and summing over all these terms. For example consider the derivative of the $\mathrm{Tr}\Big[\rho(A(t);dt)\rho^\dagger(A')\Big]$ term of

the cost function. The sum for this term can be written as



(4.3.4)

Both cost function derivative terms have an infinite sum to the left and right of the region R that need to be represented. To compute this infinite sum, we utilise the pseudoinverse as outlined in Ref [80]. Consider the sum of all the derivative terms to the right of the region R of $\operatorname{Tr}\left[\rho(A(t);dt)\rho^\dagger(A')\right]$



(4.3.5)

Fundamentally, this term requires evaluating the infinite sum over powers of the transfer matrix E. The pseudoinverse arises from the fact that the transfer matrix can be broken down into a regularised term $\tilde{E}$ and a projector term P onto the fixed points l and r such that

$$\sum_{n=0}^{\infty} E^n = \sum_{n=0}^{\infty} \tilde{E}^n + \sum_{n=0}^{\infty} P^n. \tag{4.3.6}$$

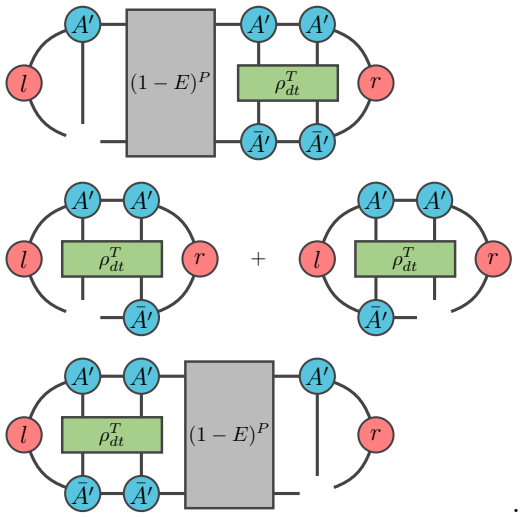
The projector term $\sum_n P^n = P$ may be divergent but does not contribute to the derivative; hence, we can project out this fixed point term. As the spectral norm of $\tilde{E}$ is less than 1, the geometric sum can be written as

$$\sum_{n=0}^{\infty} \tilde{E}^n = (1-\tilde{E})^{-1} = (1-E)^P. \tag{4.3.7}$$

In this equation, the second term refers to the pseudoinverse operator where the fixed point of E is first projected out and then the inverse is taken. Note that direct inverse calculation is not necessary numerically; instead, one can set up generalised eigenvalue equations to construct representations of the pseudoinverse for computing derivatives.

Given the use of pseudoinverses to represent the infinite sum, we can now write the full derivative for the cost function. First we consider $\mathrm{Tr}\left[\rho(A(t);dt)\rho^\dagger(A')\right]$ for $L = 2$ that has two derivative terms from the region R and two further gradient terms from the infinite sum to the left and right of the patch. Hence, the overall derivative for this term in the cost function is given diagrammatically as

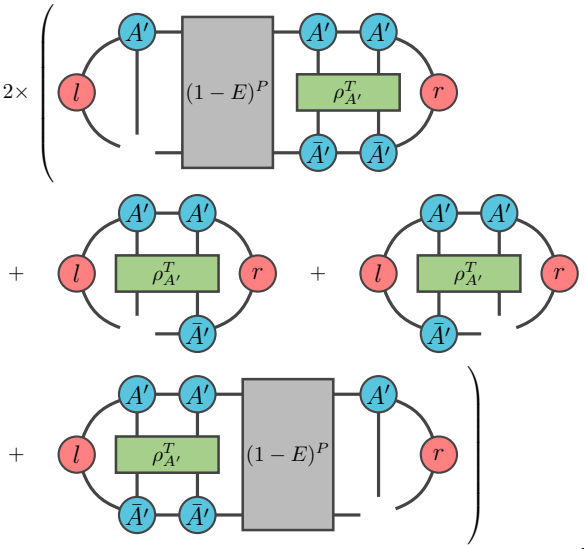
$\frac{\partial}{\partial A'}\,\mathrm{Tr}\,\rho(A(t);dt)\rho^\dagger(A') =$



. (4.3.8)

The diagrammatic representation for the term $\mathrm{Tr}\left[\rho(A')\rho^\dagger(A')\right]$ can also be constructed and is given by

$\frac{\partial}{\partial A'}\,\mathrm{Tr}\,\rho(A')\rho^\dagger(A') = 2\times$



, (4.3.9)

where $\rho'_A = \rho(A')$.

4.3.2 Optimising the local density matrix cost

Since we can now construct derivatives, we can apply gradient descent to optimise the classical local density matrix cost function. At its simplest, the updated tensor A' is given by $A' = A + \mu D$ where A is the original tensor, μ is some learning rate and D is the derivative tensor. We refer to this direct update scheme as the uniform update.

However, we would like to preserve the isometric form of the iMPS when optimising. This can be achieved by performing a polar decomposition on A' once the uniform update is applied. This method effectively finds the nearest isometric tensor to A' after performing an arbitrary update. However, this method can be unstable and slow to converge. To improve this technique, we apply ideas from the Riemannian optimisation of isometric tensor networks [107]. In Riemannian optimisation, we calculate a gradient tensor from the derivative that is used to update A whilst (approximately) preserving its canonical form throughout the update. In other words, the derivative calculates a gradient that remains in the tangent space of the isometric tensor A.

The manifold of isometric MPS tensors is the Grassmann manifold. This manifold comprises of all state tensors A that satisfies the isometric condition $A^\dagger A = \mathbb{I}$ and are invariant under the gauge AU where U is an arbitrary unitary. The following discussion summarises the necessary constructions for Riemannian gradient descent on the Grassmann manifold as applied to the MPS tensor A. For a full review of Riemannian gradient descent methods applied to isometric tensor networks, see Ref [107].

The first step in Riemannian optimisation is to map the derivative D into a gradient on the tangent space G. This projection is applied using a Euclidean metric and results in the following mapping between the derivative D and the gradient G,

$$G = D - AA^\dagger D. \tag{4.3.10}$$

Given a gradient tensor G, we can perform a retraction to update the state tensor A. A retraction is a generalised way of updating the matrix A in the direction of a tangent vector whilst preserving the properties of the manifold. The choice of retraction is not unique, and several exist for the Grassmann manifold [107]. Given a tangent vector $X = \mu G$, where μ is a tunable hyperparameter equivalent to step size, we use the following retraction.

$$R_A(X) = \begin{pmatrix} A & V \end{pmatrix} \exp \begin{pmatrix} 0 & -X^\dagger \\ X & 0 \end{pmatrix}. \tag{4.3.11}$$

In this equation, V is a unitary completion of A.

We now have all the components required to perform Riemannian gradient descent. Given an initial tensor A, we first calculate the projection of the partial derivative D to calculate the gradient G within the tangent space. We then use the above retraction to calculate the updated tensor for the state at the next optimisation step. Although this is out of the scope of the current work, one can also construct vector transport equations to use information about gradient tensors from previous optimisation steps. This information is crucial for more sophisticated optimisation algorithms like conjugate gradient methods.

The efficiency of this algorithm can be improved by applying suitable preconditioning to the gradient tensor G before performing the retraction. In MPS literature, it is common to use a preconditioner as follows

$$G \to G(r + ||G||^2 \mathbb{I})^{-1}, \tag{4.3.12}$$

where r is the right fixed point of the transfer matrix. In this instance, the preconditioner acts as a regulariser. When this preconditioner is applied in the context of Riemannian optimisation of MPSs it has links to imaginary time evolution using TDVP [107].

To test the impact of applying Riemannian gradient descent, we consider the speed of convergence of an arbitrary optimisation. We utilise the local density matrix cost function on a patch of size 2 to optimise an ansatz tensor to match a random initial iMPS. Figure 4.1 shows the convergence of the trace distance between the two local density matrices over the patch and the gradient norms. These results show that the Riemannian update with no preconditioning is slower than a straightforward uniform update, with a final trace distance worse than a uniform update. This decrease in performance may be due to the additional restriction of forcing the update to remain in the canonical MPS manifold. Despite this, the gradient norm is smaller in the Riemannian update with no preconditioning case than in the uniform update after 100 iterations. However, with preconditioning, Riemannian gradient descent converges significantly quicker and to a lower minimum than alternative optimisation strategies. The final trace distance between the ansatz and target states is lower after 500 iterations for the manifold update with preconditioning compared to the uniform update.

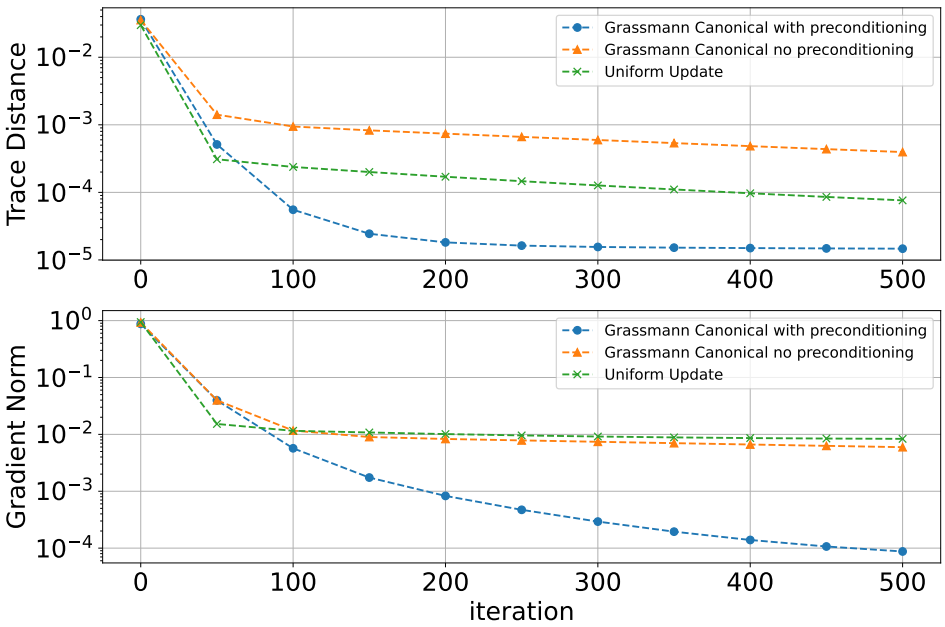


Figure 4.1: **Comparing optimisation for local density cost function:** Here, we show a trace of the optimisation at one time step using gradients calculated on the uniform MPS with no modification. In addition, we show Riemannian descent on the Grassmann manifold, with and without the preconditioner. Note that although the uniform update is effective, it does not preserve the isometric conditions of the underlying tensor network. The Riemannian descent optimisation on the Grassmann manifold with a preconditioner converges to a significantly lower trace distance from the original evolved reduced density matrix and has a significantly smaller gradient norm of the update at the final step.

4.3.3 Testing with dynamical phase transition

To test this time evolution algorithm, we benchmark it using the DQPT of the TFIM as detailed in Section 3.2. As we are interested in applying this algorithm to study the system's medium to long-time behaviour, we evolve the system up to $t = 5$. The patch size used for this experiment is $L = 4$. Note that we expect this algorithm to be more accurate as patch size increases. The optimisation algorithm used is Riemannian gradient descent using the Grassmann manifold with preconditioning. The iMPS representing the ansatz is limited to a maximum bond dimension of $D = 4$.

Figure 4.2a demonstrates that this update procedure works well for this experiment, particularly at early times, as three recurrences of the Loshcmidt echo are observed. However, at early times, some points do not entirely lie on the analytical curve, such as around $t = 1$. After $t = 6$, the results begin to diverge significantly from the analytical. This figure also compares our algorithm to iTEBD with a maximum bond dimension 190 at $t = 10$. The dynamics produced using iTEBD match the analytical curve exactly, and we will use them as a reference for the performance of our local cost function algorithm.

The Loschmidt echo curve that we use to determine the performance of this algorithm is a global measure of the state and is dependent on state fidelity. However, as our algorithm attempts to preserve the local density matrix equivalent, we prefer to construct a local measure of the Loschmidt echo, which we refer to as the trace Loschmidt echo. The trace Loschmidt echo is given by

$$\lambda_L(t) = -\frac{1}{L} \log \left(\mathrm{Tr}[\rho_N(A(t))\rho(A(0))] \right), \tag{4.3.13}$$

where $\rho_N(A(t))$ is the local density matrix with patch size L at time t. As $L \to \infty$, the trace Loschmidt echo approaches the state fidelity-based Loshcmidt echo. Figure 4.2b shows the trace Loschmidt echo at all times for iTEBD and our local cost function algorithm. The early time performance for our algorithm tracks the iTEBD trace Loschmidt echo significantly better. At late times, the performance breakdown seems to happen slightly later and in a more controlled manner, where the recurrences' shape is still present.

Figure 4.2c shows the error measured as the fidelity density with the state found using iTEBD. As this is a global measure of closeness, we expect our algorithm to deviate significantly. The deviation of fidelity density from the iTEBD curve seems to line up well with the

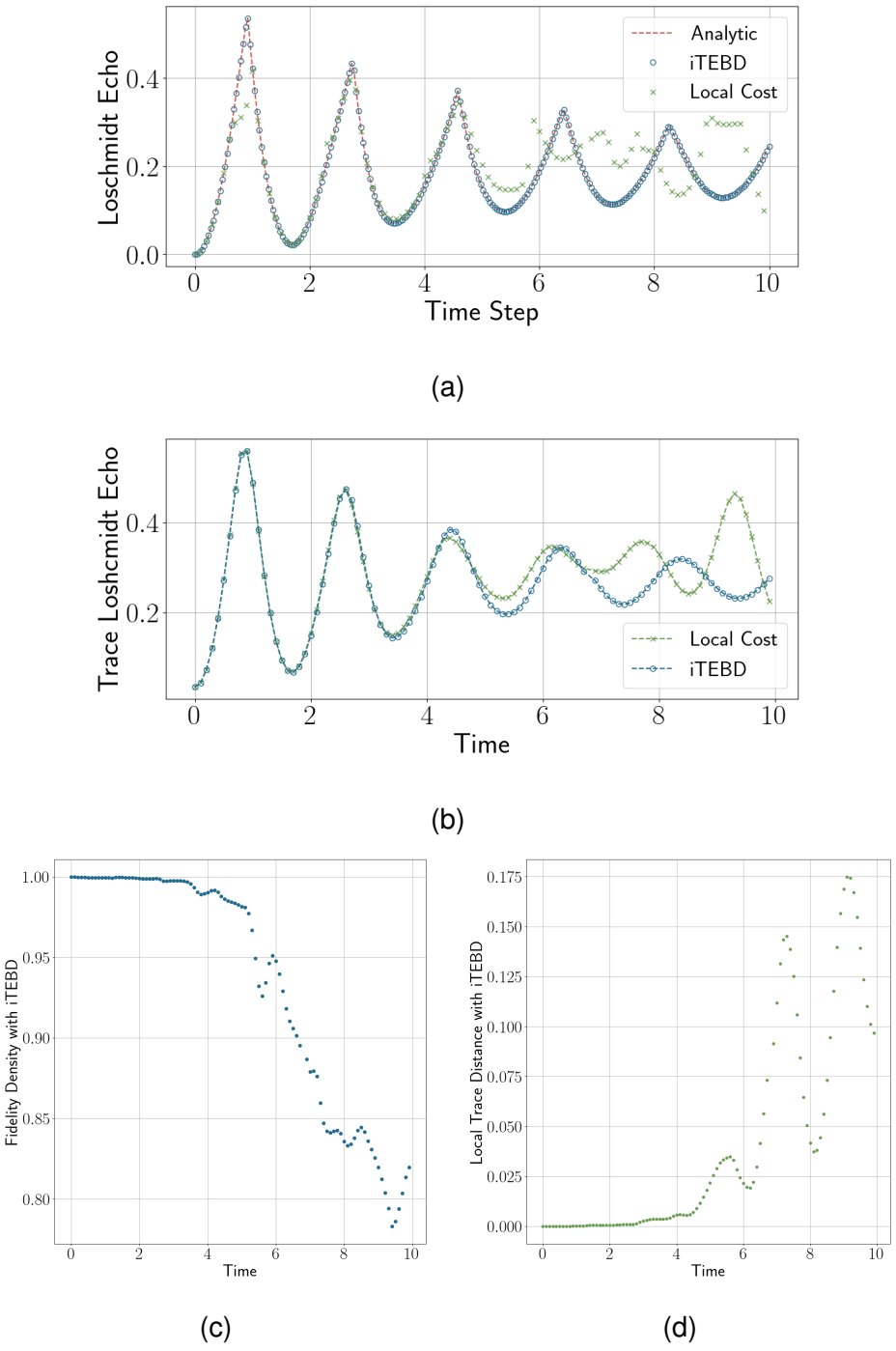


Figure 4.2: **Simulating DQPT using local density matrix cost function:** a) Shows the Loschmidt echo for the quench dynamics of the TFIM using the analytical results, iTEBD and the local density matrix evolution algorithm. The bond dimension of iTEBD was unbounded and rose to $D = 190$ by the end of the simulation. The density matrix cost function was bounded to a bond dimension of $D = 4$. b) In the same experiment, we measure the local density matrix equivalent of the Loschmidt echo, the trace Loschmidt echo given by equation 4.3.13. The local measure was constructed using a patch size of $N = 4$. c) The fidelity density of the state found using our algorithm compared to iTEBD. d) The trace distance of the reduced density matrix formed using iTEBD and our algorithm on a patch size $N = 4$.

decay in performance of the Loschmidt echo curve, with a significant decrease in fidelity at around $t = 5$. In contrast, Figure 4.2d shows the trace distance between the local density matrix found using our algorithm and iTEBD for a patch of size $N = 4$. Until time $t = 6$, this error is well controlled at order 10^{-2}, seemingly maintaining a lower value for longer times than the fidelity density. After this time, the error increases significantly, possibly due to the limited bond dimension or the support of the operators spreading beyond the patch size. This behaviour of the trace distance error may explain the divergence in the trace Loshcmidt echo performance at late times. In theory, as the patch size is increased the behaviour of the algorithm should tend towards that of iTEBD. However, if our hypothesis for developing this algorithm is correct, there will be a finite patch size beyond which local measures remain accurate throughout the evolution. Thus requiring less resources than the implied infinite patch size of iTEBD whilst still providing good understanding of the physics behind thermalisation in the long time limit.

4.4 Local density preserving circuits

Evolving the state by prioritising local density matrices can be adapted to run on a quantum device. We adjust the algorithm in Chapter 3, where we implement the local density matrix cost function instead of the fidelity density cost function. To calculate the local density matrix cost function in equation 4.3.1 requires evaluating each of the three terms in this cost as independent circuits. As an example, take the $\mathrm{Tr}[\rho(A(t); dt)\rho(A')]$ term, the circuits for this term are shown in Figure 4.3. Once again, we have flexibility in the structure of the circuits to be in the space-like or time-like form. The choice of circuit structure will depend on device performance, allowing this algorithm to be portable between different architectures.

There is a potential advantage in adapting these circuits onto a quantum device due to the advantage in evaluating the cost for large patch sizes. Growing the patch size on a quantum device linearly increases the depth and width of the circuits, even for large bond dimension. Classically, this is prohibitively expensive, aside from a relatively small range of patch sizes, as larger patches might require higher bond dimensions and an increase in the number of contractions with patch size. However, optimising these circuits on NISQ devices will likely require similar considerations to the ones used when optimising the

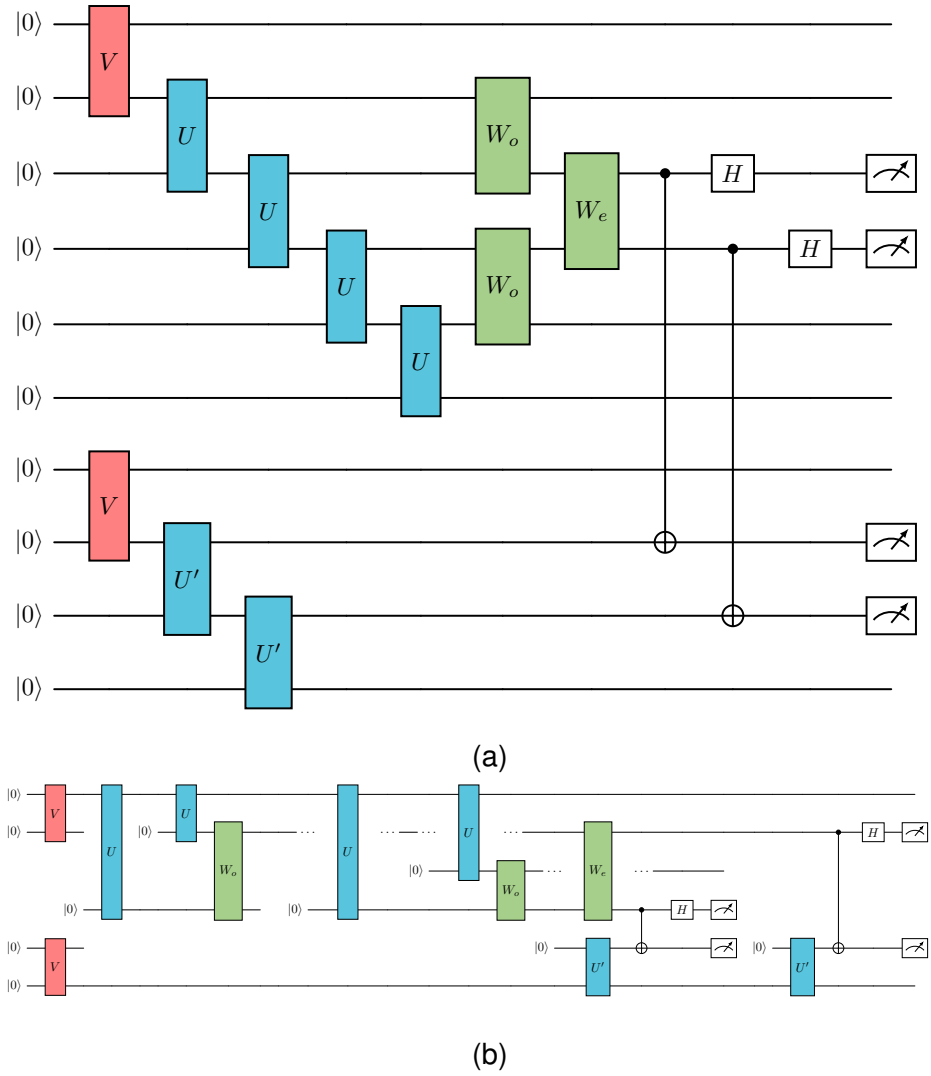


Figure 4.3: **Local density matrix cost function circuits** The a) space-like and b) time-like circuits needed to calculate the $\mathrm{Tr}[\rho(A(t); dt)\rho(A')]$ term of the local density matrix cost function. The unitary U represents the state tensor $A(t)$ and the unitary U' represents the state tensor A'. The measurements will once again be used to perform the SWAP test. The evolution operators in this figure correspond to a first-order Trotterisation. The operators W_e and W_o refer to the Trotterised evolution terms $e^{-iH_i t}$ where H_i is H_{even} and H_{odd} respectively.

fidelity density cost function. In particular, carefully considering the approximations to the left and right environments, considering potential methods to reduce the shot cost associated with optimisation and developing error mitigation techniques for NISQ devices will be necessary.

4.5 Discussion

This chapter proposes a novel algorithm for performing real-time evolution by preserving local density matrices. By optimising the cost function based on the trace distance between the local reduced density matrix, we can evolve a iMPS state ansatz through a global quench. To do so, we show how to calculate the derivatives of this cost function classically with the translationally invariant ansatz. Furthermore, we outline how to optimise this cost function so that it remains on the manifold of canonical iMPSs through Riemannian optimisation. We can reproduce the DQPT of the TFIM up to a long time with a small patch size. Furthermore, we propose quantum circuit equivalents for the classical cost function, which can be optimised to perform this evolution on a quantum device.

The proposed purpose of this algorithm is to investigate thermalising systems. Rapidly thermalising systems such as the non-integrable regime of the TFIM, as outlined in Ref [108], would be a good test of the classical algorithm. We hypothesise that this algorithm will require patch sizes independent of the system size and instead dependent on intensive quantities such as thermal correlation length. A systematic investigation of the necessary patch size to simulate a thermalising system based on intensive properties of the underlying Hamiltonian of the evolution would be a promising next step.

Varying the patch size requires improving the efficiency of constructing the derivative tensors and optimisation. The naive order of contraction outlined in this chapter may not be optimal. There are several redundant portions of contraction; therefore, caching and improving contraction ordering can significantly enhance the performance of this algorithm. Furthermore, the Riemannian optimisation algorithm outlined here is analogous to gradient descent. More sophisticated optimisation algorithms, such as conjugate gradient methods, could be used to improve the convergence rate.

In addition to the experimental development of this local density matrix cost function, there are open questions regarding its theory. Given this cost function, writing explicit equations of motion for the tensor would be beneficial. Theoretically, there should be similarities to the TDVP equations, which can be derived from the fidelity density cost function. If explicit equations can be constructed from optimising the local density matrix cost function, it would be interesting to consider its behaviour as the patch size $L \to \infty$. We hypothesise that there should be similarities between these equations and the TDVP equations for

state optimisation.

This algorithm has been designed to map the cost function to a quantum device directly. Quantum devices will likely have an advantage given the polynomial increase in the size of the circuits with patch size. Analogous to Chapter 3, implementing this algorithm on NISQ devices requires careful consideration of the optimisation strategies when applying the local density matrix cost function on a quantum device. Porting ideas such as the classical initialisation scheme, it is worth running the DQPT problem on a quantum device using the local density matrix cost function as a test. Further development is required to grow the circuit's patch size and bond dimension.

Part II

Quantum Tensor Networks for Machine Learning

Chapter 5

MPS Pre-Training

In this chapter, we propose an improved approach to the training of PQC by intialising it with a classically trained MPS. We apply this procedure to quantum machine learning (QML) to train a variational quantum classifier. In particular, we suggest that this technique effectively reduces the effects of exponentially flat cost function landscapes, otherwise known as barren plateaus. Training the classical MPS for machine learning tasks required an implementation of the DMRG inspired optimisation scheme outlined in Ref [60]. We show that initialising a general brick wall circuit with the classically trained tensor network improves accuracy and reduces loss compared to random and identity initialisations. This technique broadly applies to other variational problems which are candidate areas for quantum advantage, including quantum simulation and combinatorial optimisation, as shown in Ref [109]. However, I focus on machine learning for brevity and clarity in this section. Warm-start initialisation techniques like this are likely to be crucial for efficient optimisation on near-term hardware due to device error and hardware limitations.

This chapter begins with an introduction to variational quantum classifiers and discusses barren plateaus. Following this, I outline the classical MPS classifier algorithm outlined in Ref [60] that we use to train the MPS before embedding. I then outline the overall algorithm for initialising classically trained MPSs onto parameterised quantum circuits that we refer to as the *MPS pre-training* algorithm. Finally, we show the utility of the MPS pre-training algorithm for training a brick wall circuit to classify images from the Fashion-MNIST dataset.

This project was done in collaboration with James Dborin, Fergus Barratt, Lewis Wright and Andrew G Green. My contributions involved coding up and training the classical MPS before embedding on the quantum classifier. This work is published and forms part of Ref [109].

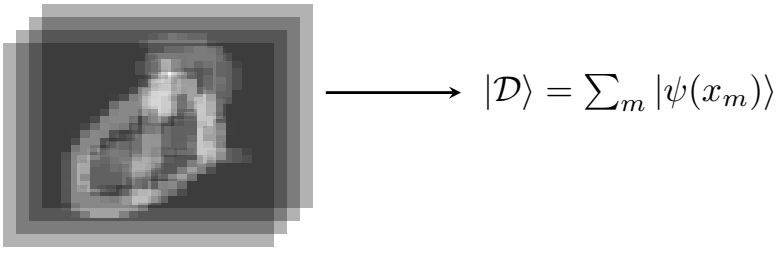
5.1 Variational quantum classifiers

Utilising quantum computers for machine learning tasks is a field of significant recent interest. QML is a broad field with several parallel research directions. For example, quantum algorithms could represent full quantum models or speed up linear algebra subroutines needed for pre-existing classical algorithms. Differentiation has been made between the performance of quantum models for quantum data and the performance possible when loading classical data. Variational quantum models - utilising PQC - stick closely to the view of machine learning models presented in algorithms such as neural networks. Here, the quantum circuits provide a model ansatz trained over variationally. Alternatively, it has been proposed that the route of encoding data in a high dimensional Hilbert space means that QML is more reminiscent of kernel-based methods. A full review of QML literature is beyond the scope of this thesis; however, Ref [110] provides a broad introduction to modern techniques in this field. This section reviews the aspects of QML most relevant to this thesis, namely training PQCs (sometimes referred to as quantum neural networks) for supervised learning.

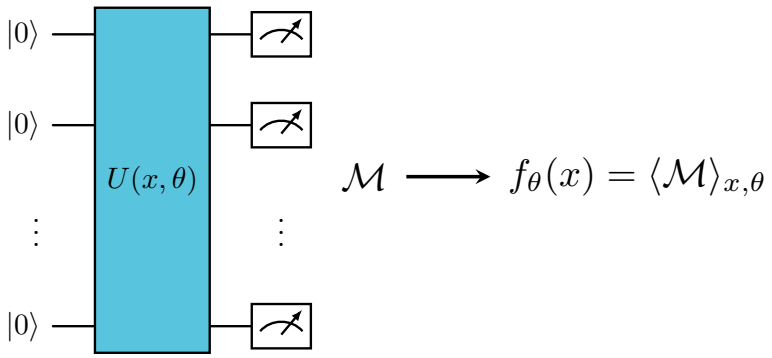
5.1.1 Overview of supervised learning

Supervised learning is one of the two main learning problems we are interested in solving in machine learning. This task has an input domain $\mathcal{X}$ and an output domain $\mathcal{Y}$. We are given a labelled dataset D, a set of N points (x_i, y_i) where y_i is the point in the output domain corresponding to x_i in the input domain. This correspondence comes from an unknown underlying probability distribution $p(y, x)$. The task in supervised learning is to generate a model for which an unclassified input $x \in \mathcal{X}$ will produce a corresponding output $y \in \mathcal{Y}$ that follows the distribution $p(x, y)$.

Generally, supervised learning is done under a restricted model family f. For example, in classical machine learning, the particular architecture of the neural network may restrict the model family. Generally, the optimal model within the family is found by minimising loss function $L(f(x), y)$. Loss measures the differences between the prediction a particular model makes $f(x)$ with the actual value y. The optimal model is defined as the one having the minimum average value of L, averaged over the entire data domain. In classical machine learning, the model family is often optimised by variationally updating model parameters θ_i.



(a)



(b)

Figure 5.1: **Overview of variational quantum classifiers:** a) Some data, whether quantum or classical, is embedded in a quantum wavefunction using a quantum data encoding. The entire dataset wavefunction $|\mathcal{D}\rangle$ is a superposition of individual data points $|\psi(x)\rangle$. b) A general quantum model $f_\theta(x)$ represented by a parametrised unitary $U(x, \theta)$. Note that in this diagram, the data embedding and model are represented by the same unitary. This quantum view allows for general protocols such as data reuploading where the data embedding is interspersed with the PQC representing the model. The quantum model $f_\theta(x)$ is trained using the variational quantum algorithm paradigm outlined in Section 1.2.3.

There is a natural correspondence between minimising the loss of classical machine learning models and optimising the cost function of PQCs. We must specify three main concepts to fully specify the translation of the classical machine learning ideas to QML. Namely, how data can be encoded, how quantum models are constructed and how the loss is optimised in the context of variational quantum algorithms. Figure 5.1 outlines the first two steps.

5.1.2 Quantum data encoding

Assuming we can represent the input data classically, it is necessary to embed the data into quantum states to process them on a quantum device. The choice of quantum feature map - the process of convert-

ing classical data into a quantum state - can heavily determine the quantum model's efficiency and performance. Several feature maps exist, including basis encoding, amplitude encoding and Hamiltonian encoding. It is also possible to train the feature map [111] to improve the performance of the quantum model. Here, we give an overview of two simple examples of quantum feature maps: basis encoding and amplitude encoding.

Basis encoding

The basis encoding is one of the simplest methods of encoding input data. It involves representing the input data point x on a binary sequence of length τ. This binary sequence is then loaded onto the qubit register. If the input x has more than one dimension, then each dimension has a specified sub-length τ_s, and the entire input bit string is constructed by concatenating the bit strings of each dimension.

For example consider an input data vector $x = (2, 7)$ encoded on a input bit string of length $\tau = 6$. We can consider each dimension of the input vector encoded onto a bit string of length $\tau_s = 3$ and concatenate the two bit strings to get the input bit string. In this case the input vector is encoded as $(2, 7) \rightarrow (|010\rangle, |111\rangle)$ and the final input state is $|010111\rangle$.

In general an input bit string $x = (b_1, b_2, \ldots, b_n)$ with $b_i \in 0, 1$ of length n requires a quantum state $|x\rangle = |b_1, b_2, \ldots, b_n\rangle$ of n qubits. For a single data point, this is readily loaded into a quantum computer using a single layer of Pauli X rotations on each qubit i where $b_i = 1$. Hence, this representation requires at most n gates and is gate efficient. Although many qubits may be needed to represent data with high precision.

The data can be loaded in superposition to represent an entire dataset $x^m \in \mathcal{D}$ with M data points. This ends up being a superposition of the basis states $|x^m\rangle$ and corresponds to

$$|\mathcal{D}\rangle = \frac{1}{\sqrt{M}} \sum_{m=1}^{M} |x^m\rangle . \tag{5.1.1}$$

The full Hilbert space of the quantum state represented by n qubits is much larger than the number of input data points M. Due to this sparsity, there are some efficient algorithms for loading data onto quantum devices [112]. In addition, quantum random access memories refer to devices that are supposed to perform these sorts of data-loading operations in parallel efficiently. However, designing and building this

hardware remains an open problem [113] with issues including the exponential scaling of resources when adding memory elements, increased circuit depths leading to issues with noise on NISQ devices and limitations around no-cloning theorems limiting readout.

Amplitude encoding

Instead of representing data on the basis of quantum states, one can also prepare an arbitrary complex input vector $x \in \mathbb{C}^N$ in the amplitude of a quantum state such that

$$|x\rangle = \sum_{i=0}^{N-1} x_i |i\rangle . \tag{5.1.2}$$

This encoding requires the input vector to be normalised, i.e. $\sum_i |x_i|^2 = 1$ and its length has to be padded such that $N = 2^n$ where n is the number of qubits.

In this instance the full dataset $\mathcal{D} = \{x^1, x^2, \ldots, x^M\}$ of length M can also be loaded in superposition such that

$$|\mathcal{D}\rangle = \frac{1}{\sqrt{M}} \sum_{m=0}^{M-1} |x^m\rangle |m\rangle . \tag{5.1.3}$$

Amplitude encoding requires a number of qubits $n \geq \log_2(NM)$. Preparing an arbitrary quantum state in the single and entire dataset case might be necessary. In general, arbitrary state preparation has a cost that is exponential in n.

5.1.3 The quantum model and training

The quantum model, as used in this thesis, defines a model family f_θ by a parameterised quantum circuits (PQC) with circuit parameters θ. In the most general case, the quantum model and data loading procedure is represented by a unitary $U(x, \theta)$. Therefore, the action of a particular quantum model for input data x can be written as

$$f_\theta(x) = \langle 0| U^\dagger(x, \theta) \mathcal{M} U(x, \theta) |0\rangle = \langle \mathcal{M} \rangle_{x,\theta} , \tag{5.1.4}$$

where $\mathcal{M}$ is some Hermitian operator defining a quantum observable. A diagrammatic representation of a quantum model is shown in figure 5.1b.

Like variational quantum algorithms, this quantum model is trained using a classical optimiser. This structure is sometimes called a quantum neural network in quantum machine learning literature. Hence, to

train this PQC requires defining a cost function that evaluates the loss function of the quantum model. A common example of a cost used in QML is one which, for a given input data point x_i, minimises the trace distance between the label y_i and the model prediction $f_\theta(x_i)$ given by

$$C(\theta) = \sum_i [y^i - \langle 0| U^\dagger(x,\theta) M U(x,\theta) |0\rangle]^2. \tag{5.1.5}$$

The optimisation algorithm, like in the case of PQC, is usually decided based on heuristics about device and model performance. Some of the most popular optimisation algorithms in quantum machine learning involve evaluating gradients of the cost function. It turns out that performing gradient-based optimisation using quantum models requires careful consideration due to the presence of barren plateaus.

5.1.4 Barren plateau

A challenge when utilising quantum algorithms is the question of trainability of the PQC. Examples of problems with trainability include variational training getting stuck in local minima [114, 115] or model expressibility being limited [116, 117] and hence no solution exists in the variational ansatz. A problem of significant research interest is the issue of barren plateaus [40, 118].

Fundamentally, barren plateaus refer to flat variational landscapes where changing the parameters θ in the PQC produce exponentially small changes in the cost function $C(\theta)$, and hence partial derivatives of the cost function. Mathematically, a barren plateau is defined when the variance of the cost $Var_\theta[C(\theta)] \in \mathcal{O}(b^{-n})$ where $b > 1$ [40]. As cost functions on quantum computers are evaluated using sampling, the exponentially small gradients require significant precision and hence exponential scaling in the number of shots to optimise [25]. Barren plateaus pose a significant problem in gradient-based optimisation schemes as the gradients at a randomly chosen point are vanishingly small. As gradient-free optimisation is based on evaluating the cost function at multiple points, if the loss landscape is sufficiently flat, these techniques will still have challenges optimising cost functions which display barren plateaus.

Barren plateaus arise from several different sources. Initially, they were found in the context of randomly initialised PQCs [118]. Subsequently, a wide variety of causes for barren plateaus have been found, including but not limited to ansatz expressibility [119], excessive entanglement in the PQC [37], when the cost function depends

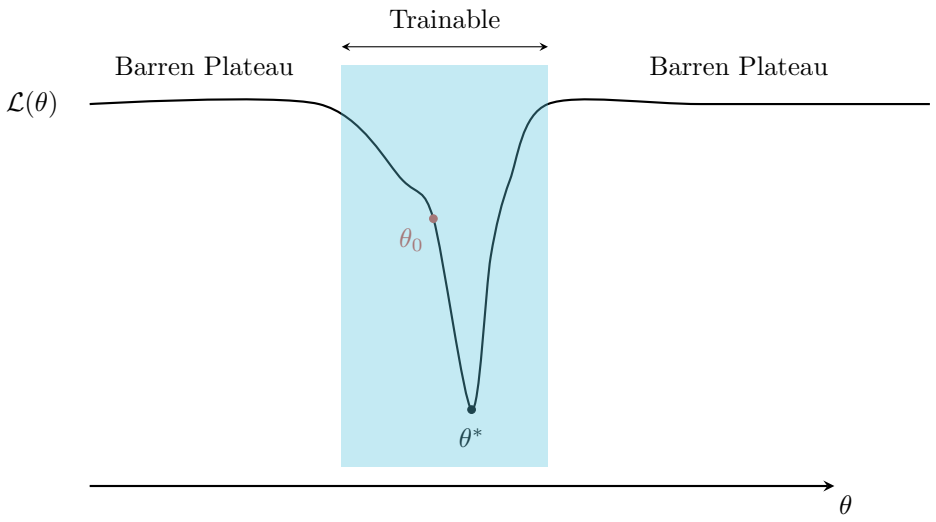


Figure 5.2: **Parameter initialisation for barren plateaus:** A schematic of a loss function $\mathcal{L}(\theta)$ which has a barren plateau. The landscape looks flat on average, as shown by the regions annotated as barren plateaus. However, some barren plateaus may have a trainable region around their minima at θ^*, such as the region highlighted in blue. The idea behind parameter initialisation is to have a method of starting the training of the optimiser with an initial point θ_0 that is within the trainable region. Hence, in this region, the gradient is not suppressed and gradient information can be used to train the model.

on a global observable [120] and Pauli noise representative of device noise if a circuit scales linearly with the number of qubits [39]. In recent years, there has been significant theoretical interest in barren plateaus, and several of the causes outlined previously have been linked to the curse of dimensionality [40]. To see this, consider a simple example where the cost function is a function of the Hilbert-Schmidt inner product $\mathrm{Tr}\left(\rho^\dagger O(\theta)\right)$. Here ρ is the input state as a density matrix, and $O(\theta) = U^\dagger(\theta) M U$ represents the represents the PQC and measurement operators. Note that any random choice of θ and a sufficiently large Hilbert space containing these operators will not align in the inner product. Hence, the inner product will be vanishingly small. Variational training schemes try to optimise functions of these inner products and, therefore, will not have sufficient information to train on unless the algorithms are tuned carefully.

Several heuristically and theoretically motivated strategies exist to avoid barren plateaus. When the problem is physically motivated or has a structure to the problem that can be utilised, restricting the ansatz is an effective way to reduce the effects of barren plateaus. For example, knowledge of a parent Hamiltonian's structure can limit the ansatz for learning mixed states [121]. One can also design an

ansatz to be shallow, thus producing an architecture resistant to barren plateaus. An example of this is the quantum convolutional network [122]. Note that shallow circuits are not enough to prevent barren plateaus. Designing the cost function limited to local terms may also be necessary to avoid barren plateaus [120] in general.

Alternatively, certain cost-function landscapes displaying barren plateaus sometimes include small regions with non-vanishing gradients around some minima. This feature is sometimes called a narrow gorge [123]. A random initialisation of parameters is likely to produce a barren plateau as it will be far from the minimum. However, initialising the parameters of the PQC close to the minima can mitigate the effects of the barren plateau. Figure 5.2 demonstrates a sketch of this strategy to avoid barren plateaus. There are various strategies to initialise the parametrisation such as by initialising blocks of the circuit to close to the identity [124], or pre-training the parameters using a classical neural network [121]. In a similar vein, training the parametrised quantum circuit in layers allows for increasing the depth of the PQC whilst remaining close to the minima in a controlled way [125]. Note that the technique of initialising the time-evolution using a simple classical model we use in Chapter 3 could be viewed as an example of this. This chapter outlines a strategy inspired by this layerwise learning approach to mitigating barren plateaus whilst utilising the MPS quantum circuit ansatz.

5.2 Tensor network machine learning

Quantum machine learning also studies classical techniques motivated by ideas from the quantum community, often referred to as quantum-inspired techniques [110]. One example of such a technique is the development of tensor network machine learning algorithms. In these algorithms, tensor networks of various architectures are used as a model family, which is optimised to perform a particular machine learning task [60, 61, 126]. Different classical optimisation techniques are also applied to these algorithms, including gradient descent-based methods that are applied to neural networks and DMRG inspired methods native to tensor networks [126].

A full review of all available tensor network machine learning methods is beyond the scope of this thesis. However, in this section, we outline one of the first algorithms used to perform supervised learning in tensor network machine learning as outlined by Ref [60]. This algorithm for performing supervised learning classically using a MPS

ansatz forms the basis of the training of the classical MPS in the pre-training work. The algorithm proposed consists of three main steps: a data encoding step to encode classical image data, a model evaluation step to process the encoded data through the model and a DMRG inspired optimisation algorithm.

5.2.1 Data encoding

As in the case of loading classical data onto a quantum computer, the data vector is encoded onto a quantum state to be processed by the MPS classifier. A product state encoding is proposed, where the input data $x = [x_1, x_2, \ldots, x_N]$ is encoded onto a state $\Phi(x) = \phi_1(x_1) \otimes \phi_2(x_2) \otimes \ldots \phi_N(x_N)$. Here, the state Φ is a d^N-dimensional Hilbert space comprised of local d dimensional spaces represented by the local embedding ϕ_i. As we are working with spin 1/2 systems, we set $d = 2$. This feature map is reminiscent of the amplitude encoding outlined in section 5.1.2. The local feature map proposed in Ref [60] is given by

$$\phi_j(x_j) = \left[\cos\left(\frac{\pi}{2}x_j\right), \sin\left(\frac{\pi}{2}x_j\right)\right]. \tag{5.2.1}$$

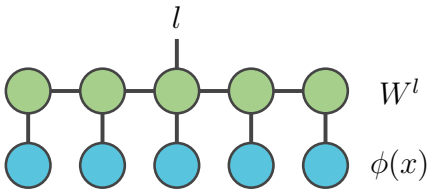
5.2.2 Evaluating the model

To classify a particular instance of an encoded input vector $\Phi(x)$, we contract it with the model represented by W by evaluating $|W \cdot \Phi(x)|$. In the case where there are two classes A and B, the contraction evaluates the probability of being in one of the classes; for example, the probability of A is $P(A) = |W \cdot \Phi(x)|$, and the probability of B is $P(B) = 1 - P(A)$. The bond dimension of the MPS model W determines the expressibility of the model.

If there are $l > 2$ classes, this algorithm employs a one-versus-all strategy to determine the class. In this view, l classifiers are trained, one for each class, and the classifier with the largest contraction with $\Phi(x)$ determines the model's prediction label. A brief notation for representing this is to turn the MPS classifier into an MPO with a single output leg of dimensionality l, represented by W^l. In the diagrammatic notation, the contraction of W^l with the encoded input vector $\Phi(x)$ is

represented by

$$f^l(x) = W^l \cdot \phi(x) =$$



. (5.2.2)

5.2.3 DMRG-inspired optimisation

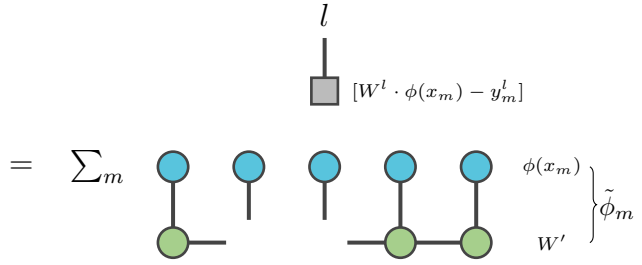
To train the MPS classifier, Ref [60] proposed a DMRG-inspired optimisation algorithm. The cost function that motivates this optimisation is the quadratic cost function given by

$$C = \frac{1}{2}\sum_m^M \sum_l (W^l \cdot \Phi(x_m) - y_m^l)^2. \tag{5.2.3}$$

Note that the sum over m is over the M input data vectors, and y_m^l represents the one hot encoded label for the input vector x_m. One hot encoding refers to the fact that when l is the correct label for x_m then $y_m^l = 1$, otherwise $y_m^l = 0$. This quadratic loss is analogous to the quantum machine learning cost function outlined in equation 5.1.5.

The optimisation algorithm works similarly to 2-site DMRG, optimising two adjacent sites simultaneously. Consider optimising the sites j and $j+1$, and the first step is to calculate the gradient of the cost function concerning this combined tensor. Consider combining the tensors A_j^l and A_{j+1} at site j - note in this instance it is assumed that site j also holds the label leg l. Differentiating the cost function with the combined vector B results in a gradient tensor ΔB^l given by the diagram

$$\Delta B^l = \sum_m [W^l \cdot \phi(x_m) - y_m^l]\tilde{\phi}_m$$



. (5.2.4)

Note that the second line of this equation represents ΔB^l where $j = 1$. The $\tilde{\Phi}(x)$ in the first line refers to an effective projection of the input vector and the differential of W^l with respect to B.

The next step in the optimisation is to perform a gradient descent update. Hence, the tensor B^l is replaced by a tensor $B'^l = B^l + \mu \Delta B^l$, where μ is a small step size. After the update, the updated model must be put back into the MPO form. In addition, the label leg l has to be swapped with the following site so that this optimisation algorithm can sweep correctly across the length of the model. These objectives can be achieved simultaneously by performing an SVD with the label leg grouped with the $j+1$ leg. Specifically, the updated tensor is broken down using an SVD such that $B'^l_{j,j+1} = \sum_{i,k} U_{j,i} S_{i,k} V^l_{k,j+1}$ and the updated tensors at site j and $j+1$ are $A'_j = U_j$ and $A'^l_{j+1} = SV^l_{j+1}$ respectively. As with the usual 2-site DMRG, truncation of the diagonal matrix S allows one to control the bond dimension of the model. In addition, this optimisation algorithm involves sweeping along the model and moving the label as necessary.

In subsequent work, the optimisation algorithm was modified to fit with traditional machine learning optimisation frameworks using stochastic gradient descent. However, we implement the DMRG-inspired optimisation scheme for this work.

5.3 MPS pre-training a quantum classifier

The MPS pre-training algorithm we propose fits into a class of heuristic warm start techniques to avoid barren plateaus. Such techniques use the fact that although on average the cost function landscape of a particular PQC may exhibit barren plateaus, there may be regions where the variance of the cost function landscape does not decay exponentially. This may be the case for areas around the minima of the cost function landscape. Therefore, initialising the parameters in the optimisation close to the minima is one way to mitigate barren plateaus and speed up training efficiency. Figure 5.2 demonstrates a sketch of this procedure.

Using the fact that MPS of a (small) finite bond dimension can be trained efficiently, we can train a classical MPS to initialise the parameters of a PQC. In a method we call *MPS pre-training*, we first train an MPS of a fixed bond dimension classically. Then, we compile this trained MPS onto a PQC by using the translation of MPSs onto a staircase quantum circuit. Finally, we continue to generalise the initialised PQC to a brick wall circuit and continue its training. This way, initialising the brick wall circuit allows us to significantly increase the speed of training the PQC.

In Ref [127], we demonstrate the flexibility of this initialisation paradigm for various settings. We demonstrate this algorithm for quantum simulation through the initialisation of VQE [43, 128]. To gain the initial MPS state, in this case, we utilise imaginary time TDVP to prepare the ground state. Additionally, we demonstrate that this technique works in optimisation problems by training it to solve Max-Cut optimisation [129]. The initial MPS is generated here by encoding the optimisation problem onto a Hamiltonian and finding the ground state of this Hamiltonian using imaginary time TEBD. In this chapter, I focus on the utility of MPS pre-training when solving machine learning problems. We focus on the binary classification of t-shirts and trousers from the Fashion-MNIST dataset [130], a general image classification problem used in machine learning.

5.3.1 Training MPS

To prepare the initial MPS for classifying Fashion-MNIST data, we utilise the algorithm outlined in Section 5.2. Before training the MPS, the data size must be reduced to fit on a circuit with a few qubits. Fashion-MNIST data is a 28×28 pixel image, with each pixel taking a value between 0 and 255 to denote the darkness of the pixel. To compress these images, we use principal component analysis (PCA) to find the principal components of the training data. To do so, we collate the training data set into a matrix of X. We then calculate the covariance matrix $\Sigma = X^T X$. Performing an SVD of the covariance matrix produces the principal components

$$\Sigma = UDU^\dagger, \tag{5.3.1}$$

where the columns of U correspond to the principal components. Given N qubits, the first N principal components are taken from U. For a given image in the training set x, the data point can be represented on the basis of the principal components by taking the inner product with the normalised principal component. For the jth principal component u_j the new value $\tilde{x}_j$ would be given by

$$\tilde{x}_j = \langle x, u_j \rangle . \tag{5.3.2}$$

Hence, the new $\tilde{x}$ is the input to the data encoding on an N qubit circuit. In particular, the data encoding used here was the rotation angle on a parametrised Pauli R_y gate for each qubit. We use this and

the classical MPS classifier training algorithm to initialise the quantum circuit.

5.3.2 Compiling MPS and training general PQC

Once the MPS classifier is trained classically, it needs to be compiled onto a PQC. The DMRG inspired optimisation scheme means that the model remains canonical throughout the training and can, therefore, be readily embedded onto a staircase circuit. During the training, the classical MPS is restricted to bond dimension two. Therefore, only two-qubit unitaries are needed to represent each state tensor. We can efficiently compile these unitaries onto a PQC using the Cartan or KaK decomposition [76]. The circuit diagram for the KaK decomposition is shown in Figure 5.3c. To initialise with higher bond dimension MPSs may require a more general decomposition routine. Certain classes of higher bond dimension MPS can be intialised using reverse-staircase circuits [63].

Once the MPS is used to initialise a PQC, this method allows us to train a more expressive PQC. This more general circuit may have been challenging to train before this initialisation due to issues such as barren plateaus. A natural way to generalise the staircase circuit produced by the MPS compilation is to embed it in the diagonal of a brick wall circuit. Brick wall circuits are general parametrised quantum circuits restricted to nearest-neighbour interactions. Hence, the diagonal of the brick wall circuit is initialised to the parameters of the MPS compilation, and the off-diagonal elements can be initialised to the identity. In doing so, the PQC has performance at least as good as that of the embedded MPS at the start of the training. Then, the optimiser is free to train the angles on all of the gates in the circuit and should converge within a relatively low number of updates. In the next section, we show this for the case of classifying Fashion-MNIST data.

5.4 Results - classifying Fashion-MNIST

To verify the MPS pre-training procedure with the binary Fashion-MNIST dataset, we compare the performance of initialising the optimisation using an MPS to random initialisation and identity initialisation. Identity initialisation is a common technique [124, 125] where some subset of the rotation angles inside a circuit are set close to zero. As mentioned in section 5.3.1, the input data is encoded using a $R_y(\theta)$ rota-

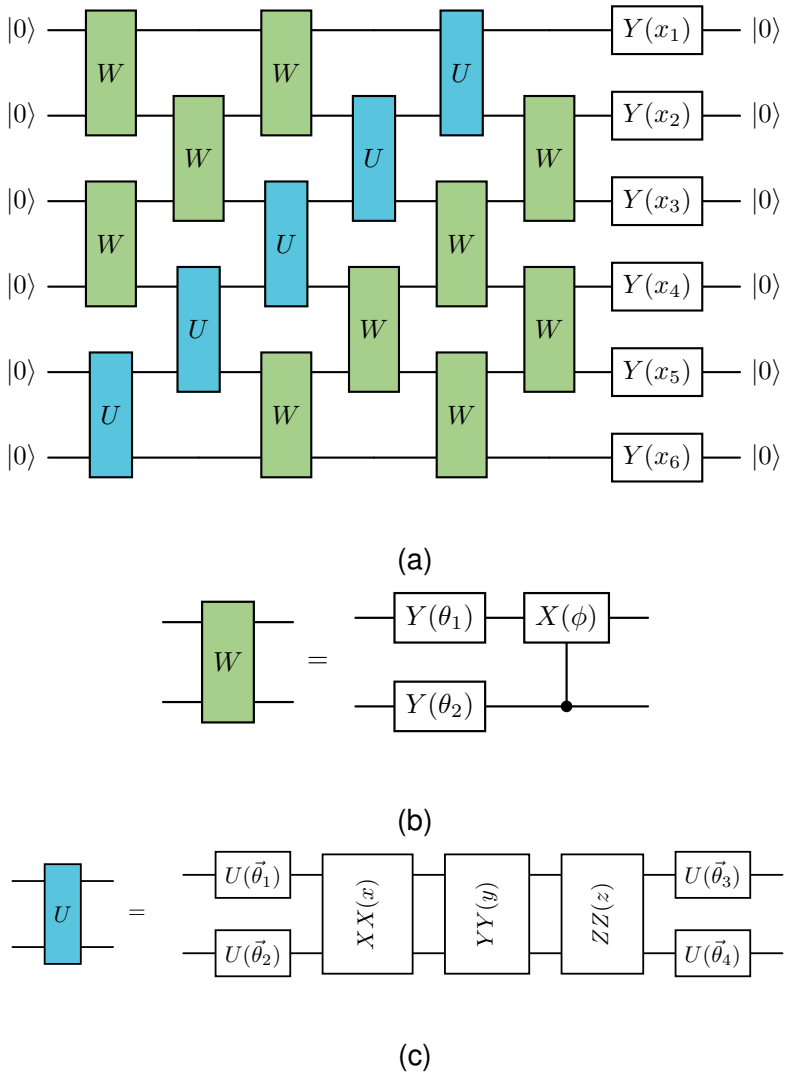


Figure 5.3: **Quantum circuits for MPS pre-training:** a) The circuit model used to train the quantum classifier for Fashion-MNIST. The classically trained MPS is embedded onto the diagonal of the brick wall using the U unitaries shown in blue. The unitaries needed to embed a classical MPS of bond dimension two span two qubits. Therefore the KaK decomposition [76] is used to embed these unitaries, as shown in c). The unitaries $U(\vec{\theta})$ in the decomposition correspond to an arbitrary single-qubit rotation parametrised by three angles. The $XX(x)$, $YY(y)$ and $ZZ(z)$ correspond to Pauli parity gates raised to a power given by the input. The off-diagonal given by the W unitaries are initialised to the identity and subsequently trained after the classical MPS is embedded. The $Y(\theta)$ unitary is a Pauli σ_y rotation gate with angle θ. The controlled $X(\theta)$ is a controlled Pauli σ_x rotation gate. The data vector $\vec{x}$ is embedded onto the circuit using $Y(x_i)$ gates on the ith qubit. The component x_i is given by the projection onto the ith principal axis of the training data. The decision function $f(\vec{x})$ is given by the probability of measuring all 0s. The label 0 is assigned if $f(\vec{x}) < 0.5$; otherwise, the label one is assigned.

tion gate on each qubit with the rotation angle θ given by each element of the input vector x. The parametrisation of the circuit representing the quantum model is given in Figure 5.3. The diagonal unitaries in the brick wall represent the MPS and the parameters are found by the KaK decomposition, and the off-diagonal unitaries are parametrised by two $R_y(\theta)$ rotation gates and a controlled $R_x(\theta)$ rotation gate. As this is a binary classification problem, the output of the model $f_\theta(x)$ given an input x is either 0 or 1. Therefore, the probability of the all $|0\rangle$ state gives $f_\theta(x)$. Hence, if $f_\theta(x) < 0.5$, then the data x is assigned class 0 or else it is assigned a class of 1.

To train the PQCs, we utilise the binary cross-entropy loss, which is given by

$$C(\theta) = -\frac{1}{N}\sum_{i=1}^{N} y_i \log[f_\theta(x_i)] + (1-y_i)\log[1-f_\theta(x_i)], \tag{5.4.1}$$

where x_i and y_i are the ith input and label data from a dataset of size N. Noiseless simulation is used to train the PQC, which is done using the Adam optimiser. Training is done stochastically in batches of the input data, and an epoch is defined as when one full run of the entire dataset occurs.

Figure 5.4 shows the results from training the PQC using random initialisation, identity initialisation and MPS pre-training. Note that the MPS initialisation used five epochs of classical training before compiling on the PQC. The classical pre-training results in an initial MPS with an accuracy of $70-90\%$. The results show that the MPS pre-training provided a lower loss and higher accuracy throughout the optimisation. In particular, at the early stages of the optimisation, the MPS pre-training worked significantly better, only taking two further epochs of training to reach a plateau compared to the other methods that required at least four epochs. Early epochs show a much better performance in the MPS pre-trained circuits, showing that the circuits are initialised much closer to the optimal point in the cost function landscape. Hence, this technique may be resistant to barren plateaus in certain instances.

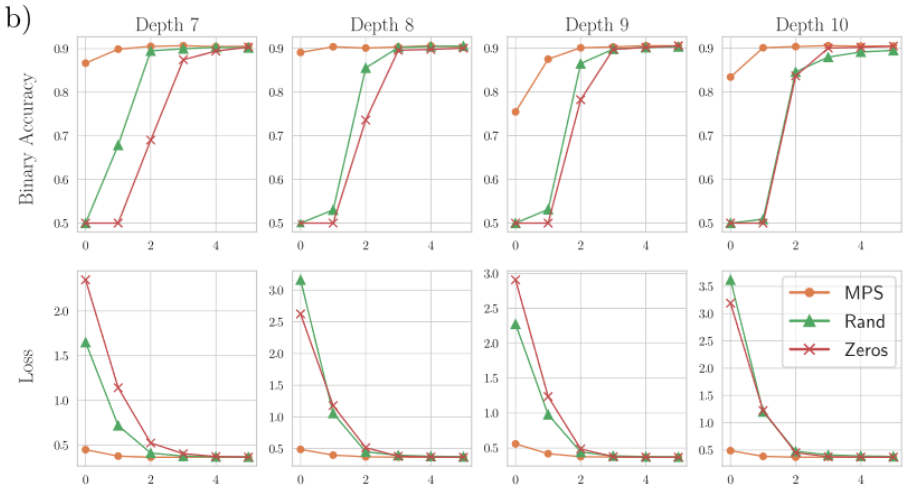


Figure 5.4: **Results from MPS pre-training:** The binary accuracy and loss is given for training a PQC using random initialisation, zero initialisation and MPS pre-training. These are given as a function of the training epoch to determine the strategy that converges the most efficiently. The classical MPS was trained for five epochs before being compiled onto the quantum circuit as in Figure 5.3. Pre-training the circuit using an MPS allows the training of the PQC to converge within two epochs. Compared to the other methods requiring at least four epochs for the training to converge, MPS pre-training is more efficient on the quantum device. There is a tradeoff being made here where additional classical resources are needed for training. The idea is that using classical resources on a smaller model can initialise a more complex PQC whilst avoiding a barren plateau. The MPS pre-trained circuits also start the training at a much lower loss and higher binary accuracy, suggesting they are in a region of the loss landscape that is easier to train. To confirm this hypothesis, however, would require training on a problem which clearly shows a barren plateau.

5.5 Discussion

This chapter outlines a method that utilises the efficiency of classically trained MPS to help train general PQCs. Our method initialises the variational quantum circuit in a place in the cost function landscape close to the minima of the loss, thereby allowing for efficient training. We demonstrate this technique for training a variational quantum classifier on a binary classification problem from the Fashion-MNIST dataset. We classically train the MPS using a tensor network-inspired classical training scheme. We propose that this technique effectively heuristically mitigates the effects of barren plateaus as regions around minima are likely candidate areas for non-vanishing gradients. This initialisation scheme allows for training of PQCs that might otherwise be difficult to train.

The proposed technique is similar to other techniques to mitigate the effects of barren plateaus. A broad class of warm-start techniques initialise quantum circuits to improve trainability. Other classical pre-training methods, such as neural networks, can initialise the parameters of a PQC [131]. The PQC can be sequentially grown based on insight about the physical problem, allowing for an ansatz that is restricted in its expressibility based on the problem [132]. For some PQCs, the parameters for a problem concentrate around particular values; hence, parameters can be transferred between different models [133], allowing for an effective initialisation. These warm-start techniques are known to be some of the most promising techniques for mitigating the effects of barren plateaus [40] and are reminiscent of the heuristic success that parameter initialisation has in classical machine learning literature. This technique is a general one which can be applied beyond the context of machine learning. In Ref [127], we explore how this technique can be used for problems in quantum simulation and combinatorial optimisation. Alternatively, we could use insights around warm-start techniques to improve the initialisation of the optimisation in the variational time evolution algorithm developed in Chapter 3 of this thesis.

The MPS pre-training technique has an interesting relationship with layerwise learning strategies [124, 125]. Here, a PQC is divided into layers, and one layer is trained at a time whilst the remainder stays fixed. In theory, this is very similar to the MPS pre-training scheme, as the layers of the circuit are often chosen to be a patch of up to the depth two nearest-neighbour interacting unitaries. This type of circuit can be efficiently represented by a low bond dimension MPS. Hence,

this type of training can be viewed as the training of a finite correlation length MPS. The parametrisation of the layers is often a more restrictive ansatz than the more general MPS representation. Hence, classically training and embedding the MPS may provide a better initialisation than layerwise training. The MPS pre-training technique could be readily adapted to represent layerwise learning. Classical MPS techniques allow for gradually growing the bond dimension. When viewed in the context of MPS pre-training, this corresponds to gradually increasing the size of the diagonal of a brick wall circuit.

A natural extension of this algorithm is to consider alternative tensor network architectures. Depending on the problem, alternative tensor network geometries may be more suitable. In fact, in the context of machine learning, PEPS [134], MERA [135], and tree tensor network [61] architectures have been proposed. In theory, the choice of ansatz may require motivation based on the entanglement behaviour of the underlying data. In addition to this, we only consider MPS classifiers in one canonicalisation. This choice in canonicalisation means that the depth of the circuit needed is linear in the number of qubits. However, a mixed canonical circuit reduces the depth necessary to initialise a circuit representing this MPS. This may allow the classical MPS with the same bond dimension to efficiently initialise a PQC more densely, further improving the initialisation. Overall, many directions for future study may further enhance the performance of such tensor network initialisation schemes.

Chapter 6

Deterministic Tensor Network Classifier

As shown in Chapter 5, there has been recent interest in tensor network inspired methods for machine learning. Here, we present a novel algorithm to produce a tensor network based feature extractor by summing quantum state representations of classical images. We show how this *sum state* feature extractor can be used as a classifier and calculate the performance of such a classifier on the MNIST and Fashion-MNIST datasets. This algorithm could be used as a precursor to prepare a classical MPS classifier that can be embedded into a quantum circuit for quantum machine learning.

In addition to generating this sum state classifier, we present a method to improve this classifier through ideas inspired by quantum stacking [136]. We propose introducing a stacking unitary to multiple copies of the output of the feature extractor. In doing so, this method has similarities to data reuploading in the quantum machine learning literature. In this work, we outline potential methods for initialising this stacking unitary and compare it to classical training on the output of the feature extractor.

In this chapter, I first outline the construction of the sum state feature extractor and its use as a classifier. Following this I introduce the notion of stacking and the construction and refinement of the stacking unitary. This work is done in collaboration with Lewis Wright, Brian Coyle and Andrew Green. My contributions include discussion into the development of the sum state algorithm and stacking ideas. I developed the code and ran experiments related to stacking, particularly in improving the stacking cost function. A pre-print of portions of this work is available at Ref [137].

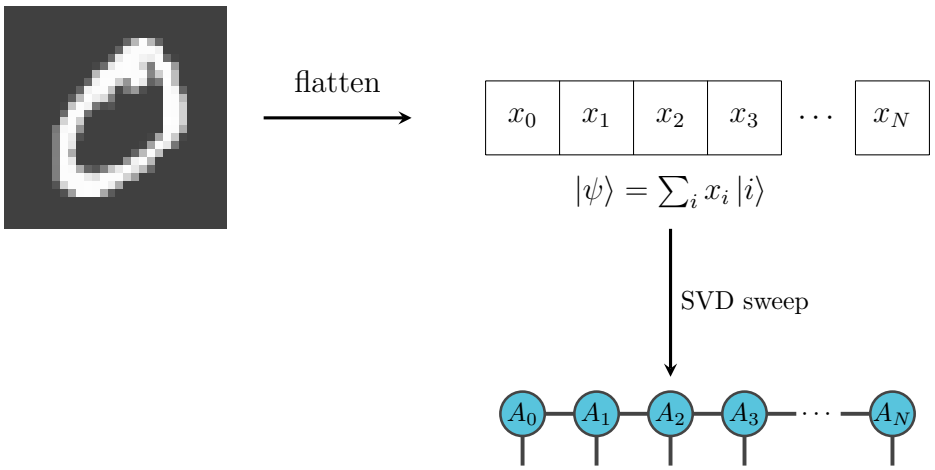


Figure 6.1: **Data encoding for deterministic classifier:** The data encoding for 2D image data first involves flattening the data by concatenating the rows along one axis. The resulting array can be viewed as the amplitude of a basis set. Through a sweep of SVDs this basis state can be turned into an MPS.

6.1 Data encoding

The deterministic generation of a tensor network feature extractor first requires encoding the data as a quantum state. As in Chapter 5, we work with image data from the MNIST and Fashion-MNIST datasets. Each image is represented by a 2-dimensional array of size $m \times m$. Each image is flattened by concatenating each row, producing a 1-dimensional vector of size $L = m^2$. The flattening of the image is equivalent to an amplitude encoding on a quantum state represented by $\log_2(L)$ qubits.

Given the data is now represented as a quantum state, we can compress the quantum state by representing it as a MPS of a fixed bond dimension. The image can be represented without compression using an MPS of bond dimension $D = \sqrt{L} = m$. Compression on an MPS state can be achieved through a sequence of SVDs whilst maintaining a fixed maximum bond dimension D_{max} as outlined in Section 1.3. Compressing using a sequence of SVDs also means that the state tensors left behind will be isometries, allowing the image MPS to be placed in canonical form. We choose to put the images in mixed canonical form about the central site where sites to the left satisfy the left canonical property and sites to the right satisfy the right canonical property. Figure 6.1 summarises the process of encoding image data onto an MPS.

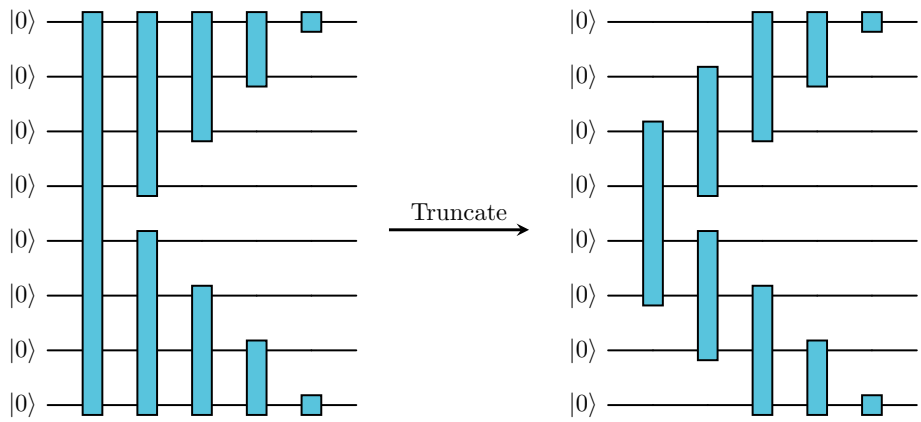


Figure 6.2: **Circuit representation of data encoding:** The mixed canonical MPS produced by the data encoding can be represented on a quantum circuit as shown above. The circuit on the left shows a fully expressive image MPS with no compressions and the circuit on the right is a compressed circuit MPS using a truncation to a bond dimension $\chi = 4$.

The fact that the final image is in mixed canonical form means that the image MPS is readily embedded on quantum circuits. Figure 6.2 shows circuit representations of the image MPS for both the full quantum state and the truncated state. Note we are using the space-like staircase representation for these circuits. In addition, since we are now working in the mixed canonical form there are two staircase circuits about the orthogonality centre. Compression or truncation of the SVD during the sweeps amounts to reducing the size of the unitaries in the circuit representation. This quantum circuit intuition is particularly useful for the refinement procedures outlined in Section 6.3.

6.2 Feature Extraction

Given the tensor network encoding of image data, we propose a deterministic method to generate a feature extractor by generating a superposition of these image states. This is done through MPS addition, firstly of images in the same class to produce a *sum state* of images in a given class. Subsequently, the sum states of each class is combined into an MPO representing the feature extractor with a single output leg. The label is encoded in the bitstring of the output leg of the MPO. The intuition behind this approach is that given enough data for a given label, the sum state for a given class is a prototypical representation of the class. In this section, we outline the construction of the sum state

for a given class, followed by the generation of the feature extractor by adding the label leg. Using this construction, we outline the process of classifying an arbitrary image using the sum state feature extractor. We also describe the effects of orthogonalising the feature extractor so that it can be embedded directly onto a quantum circuit.

6.2.1 Generating the sum state

To construct the prototypical sum state $|\Sigma_l\rangle$ for a given class l we take the unweighted sum of all the image MPSs with the class label l. The sum, in this case, is done using the standard addition of MPS. To outline this procedure, consider adding three MPSs $|A\rangle, |B\rangle$ and $|C\rangle$ given by tensors $\{a_i\}$, $\{b_i\}$, $\{c_i\}$ for $i = 1, 2, \ldots, N$, where N is the length of the MPS. For a general site at index n the sum state tensor σ_n is given by

$$\sigma_n = \begin{pmatrix} a_n & 0 & 0 \\ 0 & b_n & 0 \\ 0 & 0 & c_n \end{pmatrix}. \tag{6.2.1}$$

Note that the first and last site are special cases as they have no left and right virtual bond dimension respectively. The first sum state site is given by

$$\sigma_1 = \begin{pmatrix} a_1 & b_1 & c_1 \end{pmatrix}, \tag{6.2.2}$$

and the last sum state site is given by

$$\sigma_N = \begin{pmatrix} a_N \\ b_N \\ c_N \end{pmatrix}. \tag{6.2.3}$$

The block diagonal addition of MPS tensors produces a significant increase in the bond dimension, therefore, compression is required. This compression is achieved using sequential SVD sweeps as was used to convert the amplitude encoded image state to an MPS. Once again, if the maximum bond dimension of the SVD sweep is larger than the one required to represent the entire Hilbert space of the summed states then no compression is applied.

Creating the entire block encoded MPS sum state for a given class prior to compression is not practical for a large training set. Therefore, batches of data are taken and sequentially combined and compressed using an intermediate batching bond dimension D_{batch}. These batches are combined hierarchically until the desired sum state is reached, as

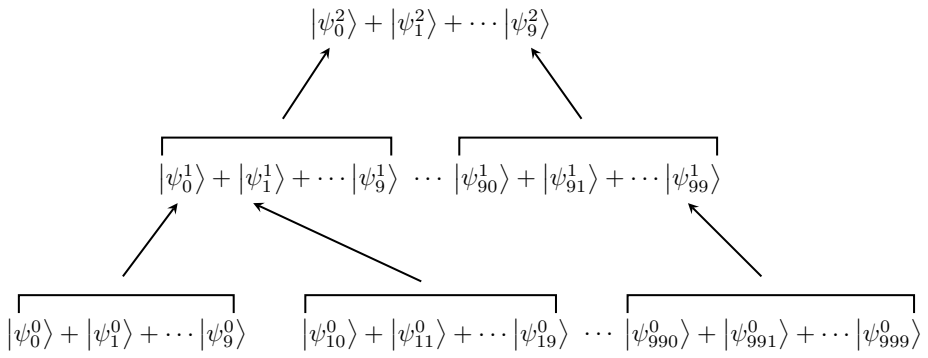


Figure 6.3: **Batching to generate sum state:** The sum state is generated in batches due to resource restrictions. This diagram shows the batching procedure applied to 1000 data points. The state $\left|\psi_i^j\right\rangle$ refers to the data point i in the j^{th} level of batching. Each data point is batched in groups of 10, requiring two layers of batching to produce the final sum state.

outlined in Figure 6.3. If D_{batch} is less than the maximal bond dimension required, then each intermediate batching involves a compression. In some cases, this may be beneficial as it regularises the images, reducing the likelihood of overfitting. This effectively performs repeated local principal component analysis at each batching layer.

6.2.2 Constructing the feature extractor

The circuit representation of a mixed canonical MPS can be interpreted as an isometric MPO acting on a reference state. This interpretation allows for the construction of a combined feature extractor from the sum state MPSs $|\Sigma_l\rangle$ where l is the class label. This is done by considering the embedding of these MPSs as MPOs acting on different reference states. By encoding the class labels on a bitstring state $|b_l\rangle$ and acting on the $|0\rangle$ reference state for the remaining sites we can draw the feature extractor as the MPO show in in Figure 6.4. Note that as we are encoding the bitstring onto qubits the number of label qubits is $\log_2(N_{classes})$ where $N_{classes}$ is the total number of classes. Hence the original sum state for a given class l can be written as

$$|\sigma_l\rangle = U_l \, |p\rangle \, |b_l\rangle \, , \tag{6.2.4}$$

where $|p\rangle = |0\rangle^{\otimes N_{pad}}$ are the padding qubits.

The sum states when constructed in this way can be readily used to classify MPS encoded images. This procedure amounts to calculating the maximal overlap of an MPS encoded image with a given bitstring.

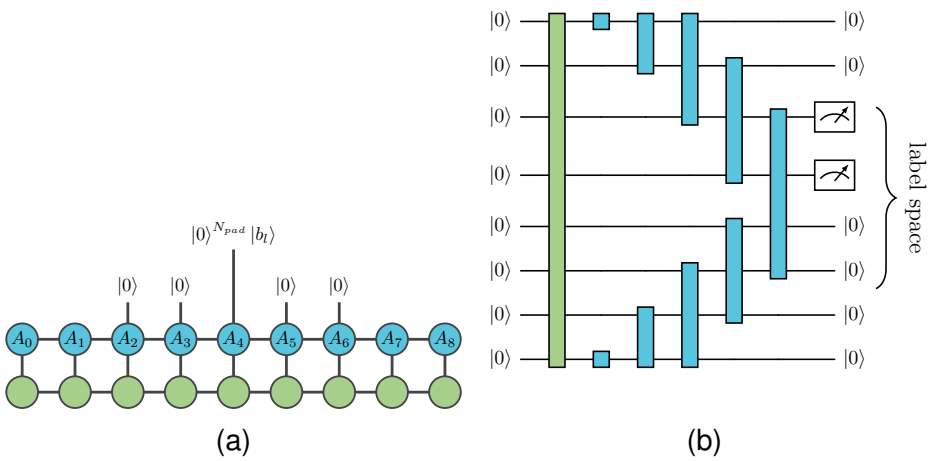


Figure 6.4: **Classification using deterministic sum state classifier:** Here, we show the sum state classifier (in blue) classifying an arbitrary image state (in green). In a) the classifier is a mixed canonical MPO with the central site containing the label space. In b), the classifier is represented as a quantum circuit.

The label k is therefore assigned to an image state $|image\rangle$ given

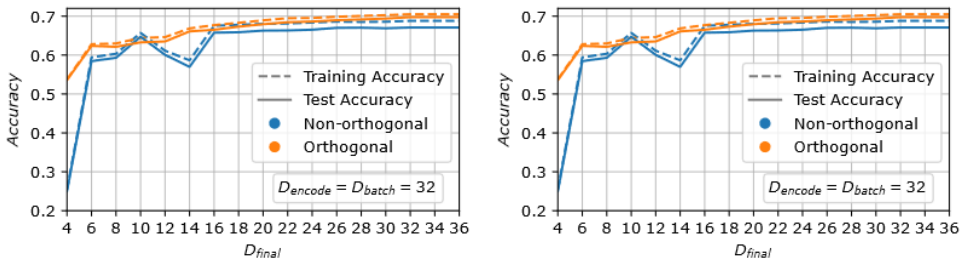
$$k = \max_l |\langle image| U_l |p\rangle |b_l\rangle|^2. \tag{6.2.5}$$

This procedure is done classically by performing the explicit tensor contractions. The feature extractor can also be directly translated onto a quantum computer. On a quantum computer the probability amplitude over each bitstring is necessary which may require significant post selection.

The individual MPO unitaries U_l can be combined and orthogonalised to form a single MPO U which represents the feature extractor. Doing so follows the same procedure as summing MPS as shown in Section 6.2.1. Note that the central site now has a class label δ therefore the combined tensor of the central site is as follows

$$\sigma_C = \begin{pmatrix} a_C^\delta & 0 & 0 \\ 0 & b_C^\delta & 0 \\ 0 & 0 & c_C^\delta \end{pmatrix}. \tag{6.2.6}$$

Hence the central site also has a different SVD procedure as it carries the class label. Due to the additional label space, the central tensor can no longer be converted to a unitary without loss of information. To achieve a unitary central site we can perform a polar decomposition, which produces the best unitary approximation to the tensor. After this summing procedure the resulting MPO can be used to classify the test



(a) (b)

Figure 6.5: **Results from deterministic classifier on Fashion-MNIST:** The training and test accuracy when applying the deterministic MPO classifier to the Fashion-MNIST dataset. Figure a) shows the results when D_{final} is varied whilst fixing D_{encode} and D_{batch} to be maximal. Figure b) Shows the accuracy when all three bond dimensions D are set equal to each other and varied. The results are shown both with and without orthogonalisation (applying a polar decomposition to the final MPO tensor). Orthogonalisation is necessary to translate the MPO onto a quantum device.

image as before, assigning a class label k to an image state $|image\rangle$ given

$$k = \max_l |\langle image| U |p\rangle |b_l\rangle|^2. \tag{6.2.7}$$

Note that instead of individual unitaries for each class U_l now we use a single unitary U. Figure 6.4 shows the explicit classification of an image using both the MPO and quantum circuit notation.

Figure 6.5 shows the results from applying this MPO feature extractor to the Fashion-MNIST dataset. The overall performance is controlled by the maximum bond dimension of the final feature extractor D_{final}. For a $D_{final} \gtrsim 20$ the test accuracy is 69.75%. However, the bond dimension of the encoded image D_{encode} and the batching bond dimension D_{batch} also constrain the performance of the feature extractor and need to be sufficiently large; for no compression the required bond dimension is $D = 32$.

6.3 Stacking Refinement

The performance of the sum state feature extractor for classification can be further improved through training. The first option to train an improved classifier is to directly improve the feature extractor unitary U. Thus, the sum state classifier is treated as an initialisation of a variational model. As this construction would involve manipulating a

large object directly, it may be challenging to train and would underutilise the insights of the deterministic scheme. Alternatively, training can be done on the output of the feature extractor, which involves a much smaller label space. Inherently, this smaller space will likely require fewer parameters to train and be more trainable. We focus on training on the output, referring to this refinement procedure as stacking. Stacking in classical machine learning literature refers to training a single classifier on the output of potentially multiple feature extractors.

At first, we focussed on training classically on the feature extractor output. To do so, we apply the feature extractor unitary U on the entire dataset and calculate the amplitude of the resulting state on each padded bitstring, thus forming the data points for a new dataset to train on. We train a neural network using the new dataset to improve the classifier performance. We use a single fully connected layer with a sigmoid activation function. This allows for sufficient non-linearity to be present within the model. To train the neural network, we utilise a categorical cross-entropy loss and apply stochastic gradient descent, training until the cost function converges. We limit the number of parameters in the model to ~ 27000 to prevent overfitting [138]. With the MNIST dataset, we show this stacking procedure works, reaching 100% training accuracy. As shown in Table 6.1 both MNIST and Fashion-MNIST show $\sim 10\%$ improvement in performance.

Similarly, we can train on the feature extractor output using a quantum circuit and the idea of data reuploading to introduce non-linearity. In this procedure we classically take multiple copies of the projection of the image state onto the label subspace after the feature extractor and encode it as the input to a larger stacking unitary V. The schematic for this procedure is outlined in Figure 6.6. The label space state $|\phi\rangle$ of a given image $|image\rangle$ is defined as

$$|\phi\rangle = \langle 0|^{\otimes N-L} U |image\rangle . \tag{6.3.1}$$

The feature extractor has a limited performance as the label space data points $|\phi_l\rangle$ and $|\phi_m\rangle$ in classes l and m respectively are not completely distinguishable as they are not necessarily orthogonal if $l \neq m$. However, we can copy (reupload) the states M times such that $|\phi_l\rangle^{\otimes M}$ and $|\phi_m\rangle^{\otimes M}$ are orthogonal if $M \to \infty$, meaning that the states can become distinguishable. The procedure of data reuploading shares similarities to multi-class amplitude amplification and is in accord with quantum classification being interpreted as a kernel method [110].

An explicit procedure is necessary to construct the stacking unitary

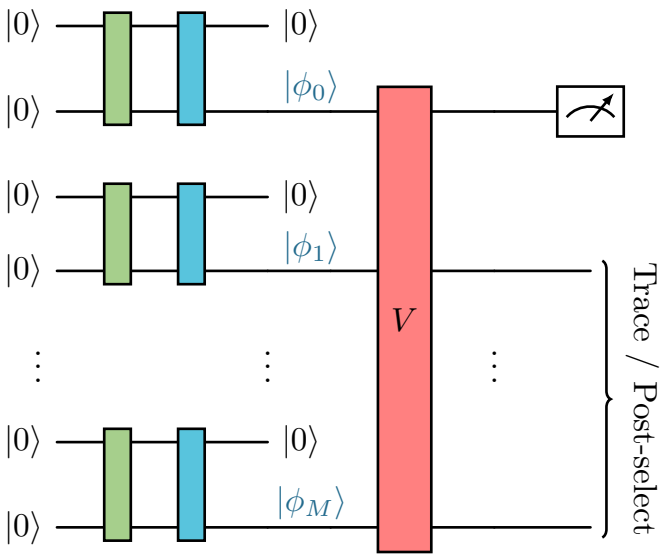


Figure 6.6: **Refining classifier using stacking:** The quantum stacking scheme is similar to date reuploading. The label space $|\psi\rangle$ is copied M times. This can be viewed as running the classification circuits shown in Figure 6.4 M times with post-selection onto the $|0\rangle$ for the padding qubits. The stacking unitary V is then applied to this copied label space. The measurement is applied to the first label space, and the remaining labels are processed by tracing or post-selection.

V. Classification under this data reuploading paradigm is performed by projecting onto the bitstring of the first label space and tracing out the remaining qubits. Hence classification occurs according to

$$k = \max_l \langle\phi|^{\otimes M} V^\dagger [|b_l\rangle \langle b_l| \otimes \mathbb{I}^{\otimes M-1}] V |\phi\rangle^{\otimes M} . \tag{6.3.2}$$

This equation can be used to construct an approximation of V. Given a perfect classification of the state using V we expect the following output

$$\delta_{l,l_\phi} = \lim_{M\to\infty} \langle\phi|^{\otimes M} V^\dagger [|b_l\rangle \langle b_l| \otimes \mathbb{I}^{\otimes M-1}] V |\phi\rangle^{\otimes M} , \tag{6.3.3}$$

where l_ϕ is the class of the state ϕ. In the limit of $M \to \infty$ the states become orthogonal and this equation can be written as

$$\sum_\phi |\phi\rangle \langle\phi|^{\otimes M} \delta_{l,l_\phi} \approx V^\dagger [|b_l\rangle \langle b_l| \otimes \mathbb{I}^{\otimes M-1}] V. \tag{6.3.4}$$

This equation can be used to construct a good initialisation for V for a finite number of copies. The matrix V is an element of the group

	MNIST		Fashion MNIST	
	Training Accuracy (%)	Test Accuracy (%)	Training Accuracy (%)	Test Accuracy (%)
Initial	78.42	80.33	73.10	71.65
Single layer neural network	100	91.07	89.39	82.60
SVM	92.10	90.46	82.47	81.14
Quantum stacking: 2 Copies	86.63	87.63	75.37	74.22
Quantum stacking: 3 Copies	90.95	89.80	76.24	74.92

Table 6.1: **Accuracies for stacking refinement:** The results from training on the outputs of the MPO classifier with $D_{total} = 32$. The classical training procedures are the single layer neural network and the SVM. The quantum stacking utilises the data reuploading scheme. Note that the classical neural network is likely to be difficult for the quantum model to compete with as it includes a significant amount of non-linearity. Therefore, the SVM is likely to be a better comparison as quantum models are known to be representative of kernel methods. In addition to this, adding layers to the classical neural network means it performs so well that this test is more of a benchmark for the quantum technique rather than a test of quantum advantage.

$SU(2^{N_{label} \times M})$, which can be broken down into a grid of $2^{N_{label}} \times 2^{N_{label}}$ matrix elements, one for each label. The factor $[\mathbb{I}^{\otimes M-1} \otimes |b_l\rangle \langle b_l|]$ means that data from a given class l only contributes to the l^{th} column in this grid. Hence each of these columns of V can be initialised by constructing the left hand side of equation 6.3.4 for each class independently. To restrict this to be unitary we take an SVD of this object for a given class l resulting in $W_l \Delta_l W_l^\dagger$. The lth column of V is then given by $V_l = \sqrt{\Delta_l} W_l^\dagger$ where only the $2^{(M-1)N_{label}}$ singular values are kept. Once the full V is constructed by concatenating the individual V_l. To orthogonalise, the entire V has a final polar decomposition applied to it.

We find that the quantum stacking procedure shows improvements over the feature extractor even for a single additional copy of the data. Table 6.1 shows the improvement in performance using the quantum stacking procedure, where 2 copies refers to a single additional copy of the data. We compare our results to the classical neural network and one obtained by a classical support vector machine (SVM), which shares a similar philosophy to quantum stacking. We were only able to achieve 3 copies of the quantum stacking due to computational limitations but despite this we see continuous improvement in the classifier performance as new copies are introduced.

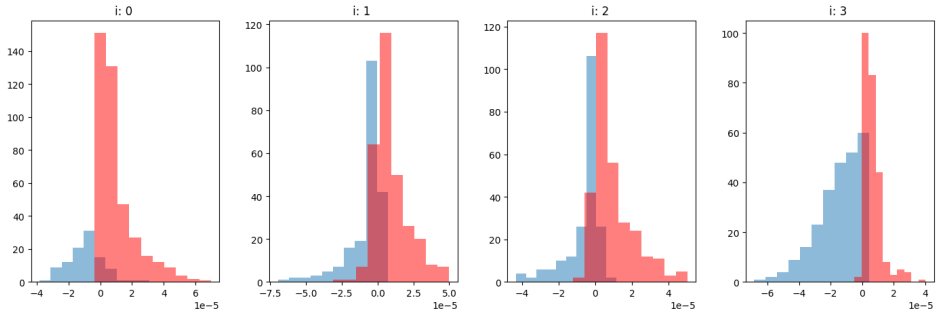


Figure 6.7: **Label space qubit histograms:** Each label space qubit i acts like a binary classifier. To investigate the boundary between the two classes at each qubit, we show the histograms for the 2 classes in each of the 4 qubits in the label space. The red histogram represents images that should project to $|0\rangle$ in the label space qubit i, and the blue histogram should project to $|1\rangle$.

6.3.1 Improving stacking cost function

Despite working to improve the quantum classifier's performance, the stacking unitary's overall performance remains significantly off the performance of both classical methods, the single layer neural network and the SVM. We propose that the performance of the stacking unitary can be improved, without additional copies, by modifying the cost function that is implicit in equation 6.3.3. Considering each label space qubit as a binary division separating two classes, the implicit stacking cost function applies a hard boundary between different bitstring classes and treats all label space qubits equally. Figure 6.7 shows that, in reality, the classes tend to overlap and the degree of overlap varies by the qubit index.

Therefore, a modified cost function could be used to find the stacking unitary that allows for a softer class boundary and can tune the boundary by the label space qubit. We propose such a cost function that we refer to as the tanh cost function and is given by

$$C = \sum_{i,j} \tanh\Big(A_j \langle\phi_i| V^\dagger P_j V |\phi_i\rangle\Big)\delta(l_i, j), \tag{6.3.5}$$

where P_j is a projection onto the $|0\rangle$ state of the label space qubit j. The term in the $\tanh$ function calculates the amplitude of this state of the j^{th} label space qubit after the action of the stacking unitary V acting on $|\phi_i\rangle$ that represents M copies of the label space for the image i. A hyperparameter A_j controls the boundary's hardness for the label space qubit; a higher value of A results in a harder boundary. Given that there is a different A_j for each label space qubit, the hardness of

the boundary can be tuned for each qubit and can be modified during training. Heuristic methods were developed to select A_j during training, namely the more overlap the classes for a given qubit, the softer the boundary.

The previous optimisation scheme to find the stacking unitary no longer works with the tanh cost function. We utilise gradient descent to optimise V to test the new cost function. The derivative D with respect to $V^\dagger$ of equation 6.3.5 is

$$D = \sum_{i,j}(1 - \tanh\Big(A_j \langle\phi_i| V^\dagger P_j V |\phi_i\rangle\Big)^2)(A_j P_j V |\phi_i\rangle \langle\phi_i|)\delta(l_i, j). \quad (6.3.6)$$

A simple gradient descent algorithm using this gradient update results in V becoming non-unitary. However, we would like to remain on the unitary manifold to effectively compare against the previous optimisation scheme. As in Chapter 4, we rely on Riemannian optimisation to update V whilst remaining on the unitary manifold. Generally, one can update a unitary by applying $V\exp(A)$ where A is anti-hermitian. We can write the derivative D in an anti-hermitian form by writing

$$A = \frac{1}{2}(V^\dagger D - D^\dagger V). \quad (6.3.7)$$

Applying the exponential at each update incurs a high numerical cost. Therefore, we consider the leading order term in the Taylor expansion of this update is $dV = VA$. Hence, the gradient descent update to the stacking unitary $V' = V + \mu dV$ given a derivative D using the Taylor expansion of a general update to a unitary as the retraction can be written as

$$dV = \frac{1}{2}(D - VD^\dagger V). \quad (6.3.8)$$

This update is unitary only up to the leading order hence after each step, a polar decomposition is applied.

Using the tanh cost function with a gradient update that preserves unitarity improves the classifier's performance. To compare, we calculate the training accuracy for Fashion-MNIST from the previous optimisation strategy to this one. With no copies, the improvements are limited, resulting in only a 0.4% improvement in training accuracy. However, for 2 copies, the improvement is more significant at a 1.1% improvement in training accuracy resulting in an overall accuracy of 76.5%. Despite this improvement, we are still significantly off the classical boundary of 82.47% produced by the SVM and 89.39% produced by the single-layer neural network. Due to the additional computational

complexity of calculating the gradients and applying the new cost function, performing this comparison for 3 copies proved too computationally expensive.

6.4 Discussion

The deterministic sum state feature extractor algorithm outlined in this chapter is a flexible algorithm for generating tensor networks to classify amplitude-encoded image data. Our algorithm is broken down into two steps. Implementing the sum state feature extractor U can be considered a feature selection step that reduces the problem's dimensionality. Subsequently, the stacking step of training on the output of the label space is reminiscent of the standard training of a classifier. The quantum version of our stacking scheme is based on data reuploading. We propose a basic quantum stacking scheme followed by a refinement that considers features of the label space of the data. There are several future directions this research could take. Note that in Ref [137], we demonstrate the flexibility of this algorithm to different tensor network architectures by applying it to train image data encoded as tree tensor network (TTN) in addition to MPS.

In addition, it is worthwhile to consider whether the type of data encoding used affects the performance of this algorithm. We used a trivial amplitude encoding to embed our data onto a quantum state. It has been shown that the choice in data encoding can significantly affect the performance of quantum machine learning algorithms [110]. The amplitude encoding we choose results in a loss of locality in the quantum state. An encoding scheme that preserves this locality in some way may perform better. In addition to modifying the encoding type, utilising a tensor network architecture such as PEPS that is specific to 2D data may be worthwhile, although it is unclear whether the additional computational effort of using a 2D network structure would provide significant improvement. A systematic review of data encoding as it relates to this algorithm and 2D tensor network architectures could be a fruitful avenue for future research.

Training using the tanh cost function is expensive. This is mainly a problem given the degree of hyperparameter tuning necessary for this cost function to be practical, given each label space qubit has a different hyperparameter A_j assigned to it. Currently, simple heuristics are used to set A_j. However, research is necessary to find a more systematic way of assigning these hyperparameters. It should be possible to parallelise the training and hyperparameter selection by utilising the fact that updates to each label space qubit can be calculated independently.

To improve the computational performance of this technique, it may be worthwhile to consider alternative ways to weigh data near the de-

cision boundary. Instead of constructing the tanh cost function directly, it may be preferable to duplicate data near the boundary with some blurring to improve the classifier's performance. This duplication can be done as a function of the tanh cost function gradient. The optimisation of the stacking unitary then follows the trivial stacking routine we initially proposed. We call this idea *sketching* and believe it to be a promising strategy for improving the training of this classifier.

Our algorithm shows that we can deterministically initialise unitaries that perform remarkably well as feature extractors and classifiers by utilising the geometric structure of the Hilbert space of quantum states. Applying these findings to improve the development of quantum machine learning algorithms could be useful more broadly. As shown in Chapter 5, this initialisation could be a good starting point for training a PQC. In addition, the core methods required to generate the sum state classifier rely on simple linear algebra subroutines such as SVD. Given that there are several quantum algorithms to perform such linear algebra tasks [83], it may be possible to design a purely quantum algorithm to implement such a sum state classifier. In particular, it is clear that scaling the quantum stacking routine is an issue. Designing a quantum native stacking algorithm may provide a route to representing larger stacking unitaries. This quantum native method could circumvent a number of the bottlenecks with variational quantum algorithm based QML techniques such as barren plateaus. Overall it is unlikely that quantum advantage is to be found with quantum algorithms acting on classical data. Therefore, investigating how to adapt the techniques outlined in this chapter towards quantum data may provide a clearer route to quantum advantage.

Chapter 7

Conclusion

This thesis introduces several connections between tensor networks and quantum algorithms capable of running on near-term quantum computers. In particular, we focus on quantum versions of the MPS. This conclusion summarises the work of this thesis and provides connections and potential future directions for the different projects.

In part one, ideas developed for MPS help us use current-generation quantum devices to simulate quantum systems in the thermodynamic limit. Initially, we focus on representing an iMPS on a quantum device, utilising the variational principle to prepare a ground state across a quantum critical point. This experiment proved that Google's Sycamore architecture can represent interesting states of quantum matter. This algorithm had not been implemented on near-term hardware prior to this work which meant that we were able to develop and quantify the impact of various error mitigation strategies and adaptations to the algorithm.

After this proof of concept, we develop algorithms towards a potential avenue for quantum utility, namely time-evolving quantum systems. Entanglement naively increases exponentially during time evolution in quantum quench experiments, quickly leading to classical techniques being unable to simulate many systems of interest. Trotterised evolution on quantum circuits requires only a linear increase in the depth of the circuits with time, hence providing a likely avenue for quantum utility. However, current-generation quantum devices have too much noise to evolve directly for a long time and are too small to overcome finite-size effects. Chapter 4 introduces a variational quantum algorithm that performs time evolution of an iMPS using a fidelity density cost function. Testing this algorithm is done using the dynamical quantum phase transition (DQPT) of the transverse field Ising model (TFIM), first by profiling the fidelity density cost function on Google's Sycamore

processor, followed by performing a full-time evolution using Quantinuum's H1 processor. From these experiments we find that particular hardware characteristics can be accommadated for by the flexibility of this algorithm. For example, Google's superconducting architecture allowed for shallower circuits utilising a high number of qubits, thus being well suited for the space-like layout of the circuit. In addition, the depolarising error on this device required the use of Loschmidt rescaling. In contrast the longer coherence times on Quantinuum's hardware allowed for the time-like rewrite, and the additional overhead of Loschmidt rescaling was not necessary. However, the limited shot budget required careful consideration of the initialisation of the optimisation.

Finally, we propose adaptations to the cost function so that the algorithm outlined in Chapter 3 preserves local information during the evolution. Developments in classical tensor network techniques inspire this approach. These algorithms preserve only necessary local information to reduce the resource requirements of simulation. However, whereas prior classical algorithms have no direct mapping to quantum circuits, our proposed algorithm can be readily adapted to prepare the local density matrix cost function on a quantum device.

We present many adaptations to the algorithm and error mitigation strategies that allow us to run on current-generation quantum device. Essential in these adaptations is the flexibility of the MPS circuits to switch between the time-like and space-like format. In addition, time evolution allows for warm-start protocols to optimise the cost function, drastically improving the sampling costs of our time evolution algorithm. However, it is clear that the resources available on current-generation devices does not allow us to push this algorithm to demonstrate quantum advantage. One aspect of the development required is better algorithms to perform more efficient optimisation with better cost functions as outlined in this thesis. However, ideally the hardware available will develop which mix the strengths of both the Google and Quantinuum device, namely better quality qubits with lower decoherence coupled with increased shot budgets.

The time evolution algorithm provides the most promising avenue towards quantum advantage. The suggested next steps for this algorithm is to adapt the local density matrix cost function onto a circuit and run this experiment on a near-term device. To push quantum advantage requires allowing for an adaptable bond dimension. Therefore, investigations into strategies for increasing the bond dimension within

the framework of the time evolution algorithm are needed. Given these improvements, a good test of this algorithm is to simulate a rapidly thermalising system. Furthermore, higher dimensional models that are more challenging to simulate classically should be considered, likely requiring adapting alternative isometric tensor networks aside from canonical MPSs. Ultimately, the goal is to simulate the evolution of a quantum system that is impractical to perform on a classical device.

The second part of this thesis demonstrates that quantum MPSs can be used outside of quantum simulation, primarily focussing on machine learning. We saw that initialising a parameterised quantum circuit improves trainability in the time evolution work. This problem of initialisation is also present in quantum machine learning. It is a standard heuristic method to avoid barren plateaus, which prevent the training of quantum machine learning models altogether. Chapter 5 presents a way that MPS circuits could initialise a more expressive quantum circuit using a small classical simulation. We applied this approach to train a classifier for the MNIST image dataset. This technique is adaptable to practically any problem that utilises parameterised quantum circuits, including the time evolution algorithms outlined in the first part of this thesis.

The training routine used to prepare the classical MPS in Chapter 5 uses a variational DMRG-inspired approach. In Chapter 6, we outline an alternative approach to training the classical MPS classifier using a deterministic procedure based on the principle of superposition to prepare a sum state. Using inspiration from data reuploading, we designed techniques to refine the performance of this classifier.

Overall, the algorithms in this thesis are a proof of concept for the utility of tensor networks as a framework for thinking about quantum algorithms running on near-term hardware. We emphasise the flexibility of this paradigm. In this thesis, the scope of study is quantum simulation and quantum machine learning. However, there are many other areas of active research where classical tensor networks are relevant, such as solving partial differential equations [62] or speeding up signal processing routines [139]. Insights from this work could be used to develop quantum algorithms for these applications. The utility of this approach is the focus that tensor networks place on entanglement as the resource of value in quantum computing. Trading insights from classical tensor networks to the design of quantum algorithms and vice versa provides the opportunity to advance both fields and provide clear boundaries between quantum and classical computing.

Bibliography

1. Chi-Chih Yao, A. *Quantum Circuit Complexity* in *Proceedings of 1993 IEEE 34th Annual Foundations of Computer Science* Proceedings of 1993 IEEE 34th Annual Foundations of Computer Science (Nov. 1993), 352–361.
2. Molina, A. & Watrous, J. Revisiting the simulation of quantum Turing machines by quantum circuits. *Proceedings of the Royal Society A* **475,** 20180767 (June 2019).
3. Bharti, K. *et al.* Noisy Intermediate-Scale Quantum Algorithms. *Rev. Mod. Phys.* **94,** 015004 (Feb. 2022).
4. Cao, N. *et al. NISQ: Error Correction, Mitigation, and Noise Simulation* Nov. 2021. arXiv: 2111.02345 [quant-ph].
5. Roffe, J. Quantum error correction: an introductory guide. *Contemporary Physics* **60,** 226–245 (2019).
6. Campbell, E. A Series of Fast-Paced Advances in Quantum Error Correction. *Nat Rev Phys* **6,** 160–161. ISSN: 2522-5820 (Mar. 2024).
7. Kim, Y. *et al.* Evidence for the utility of quantum computing before fault tolerance. *Nature* **618,** 500–505 (June 2023).
8. Moses, S. A. *et al.* A Race-Track Trapped-Ion Quantum Processor. *Phys. Rev. X* **13,** 041052 (4 Dec. 2023).
9. Evered, S. J. *et al.* High-fidelity parallel entangling gates on a neutral-atom quantum computer. *Nature* **622,** 268–272 (2023).
10. Bluvstein, D. *et al.* Logical quantum processor based on reconfigurable atom arrays. *Nature* **626,** 58–65 (2024).
11. Preskill, J. Quantum Computing in the NISQ era and beyond. *Quantum* **2,** 79. ISSN: 2521-327X (Aug. 2018).
12. Acín, A. *et al.* The Quantum Technologies Roadmap: A European Community View. *New J. Phys.* **20,** 080201. ISSN: 1367-2630 (Aug. 2018).

13. Cirac, J. I. & Zoller, P. Quantum Computations with Cold Trapped Ions. *Phys. Rev. Lett.* **74,** 4091–4094 (20 May 1995).

14. Bruzewicz, C. D., Chiaverini, J., McConnell, R. & Sage, J. M. Trapped-ion quantum computing: Progress and challenges. *Applied physics reviews* **6** (2019).

15. Strohm, T. *et al.* Ion-Based Quantum Computing Hardware: Performance and End-User Perspective. arXiv: 2405.11450 [quant-ph] (2024).

16. Pogorelov, I. *et al.* Compact Ion-Trap Quantum Computing Demonstrator. *PRX Quantum* **2,** 020343 (2 June 2021).

17. Huang, H.-L., Wu, D., Fan, D. & Zhu, X. Superconducting quantum computing: a review. *Science China Information Sciences* **63,** 1–32 (2020).

18. Koch, J. *et al.* Charge-insensitive qubit design derived from the Cooper pair box. *Physical Review AAtomic, Molecular, and Optical Physics* **76,** 042319 (2007).

19. Somoroff, A. *et al.* Millisecond Coherence in a Superconducting Qubit. *Phys. Rev. Lett.* **130,** 267001 (26 June 2023).

20. Bravyi, S., Dial, O., Gambetta, J. M., Gil, D. & Nazario, Z. The future of quantum computing with superconducting qubits. *Journal of Applied Physics* **132** (2022).

21. Zhong, H.-S. *et al.* Quantum Computational Advantage Using Photons. *Science* **370,** 1460–1463 (Dec. 2020).

22. Arute, F. *et al.* Quantum supremacy using a programmable superconducting processor. *Nature* **574,** 505–510 (Oct. 2019).

23. Pan, F. & Zhang, P. Simulation of Quantum Circuits Using the Big-Batch Tensor Network Method. *Phys. Rev. Lett.* **128,** 030501 (Jan. 2022).

24. Tindall, J., Fishman, M., Stoudenmire, E. M. & Sels, D. Efficient Tensor Network Simulation of IBM's Eagle Kicked Ising Experiment. *PRX Quantum* **5,** 010308 (1 Jan. 2024).

25. Cerezo, M. *et al.* Variational quantum algorithms. *Nature Reviews Physics* **3,** 625–644 (Aug. 2021).

26. Bravyi, S. & Kitaev, A. Fermionic Quantum Computation. *Annals of Physics* **298,** 210–226. ISSN: 00034916 (May 2002).

27. Choquette, A. *et al.* Quantum-optimal-control-inspired ansatz for variational quantum algorithms. *Physical Review Research* **3.** ISSN: 2643-1564 (May 2021).

28. Bravo-Prieto, C., Lumbreras-Zarapico, J., Tagliacozzo, L. & Latorre, J. I. Scaling of variational quantum circuit depth for condensed matter systems. *Quantum* **4,** 272. ISSN: 2521-327X (May 2020).

29. Nakaji, K. & Yamamoto, N. Expressibility of the alternating layered ansatz for quantum computation. *Quantum* **5,** 434. ISSN: 2521-327X (Apr. 2021).

30. Boyd, S. & Vandenberghe, L. *Convex Optimization* (Cambridge University Press, 2004).

31. Mitarai, K., Negoro, M., Kitagawa, M. & Fujii, K. Quantum circuit learning. *Phys. Rev. A* **98,** 032309 (Sept. 2018).

32. Schuld, M., Bergholm, V., Gogolin, C., Izaac, J. & Killoran, N. Evaluating Analytic Gradients on Quantum Hardware. *Phys. Rev. A* **99,** 032331 (Mar. 2019).

33. Platt, J. *Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines* tech. rep. MSR-TR-98-14 (Microsoft, Apr. 1998).

34. Ostaszewski, M., Grant, E. & Benedetti, M. Structure Optimization for Parameterized Quantum Circuits. *Quantum* **5,** 391 (Jan. 2021).

35. Shaffer, R., Kocia, L. & Sarovar, M. Surrogate-based optimization for variational quantum algorithms. *Physical Review A* **107** (Mar. 2023).

36. Spall, J. C. An overview of the simultaneous perturbation method for efficient optimization. *Johns Hopkins Apl Technical Digest* **19,** 482–492 (1998).

37. Ortiz Marrero, C., Kieferová, M. & Wiebe, N. Entanglement-Induced Barren Plateaus. *PRX Quantum* **2,** 040316 (Oct. 2021).

38. Sack, S. H., Medina, R. A., Michailidis, A. A., Kueng, R. & Serbyn, M. Avoiding Barren Plateaus Using Classical Shadows. *PRX Quantum* **3,** 020365 (2 June 2022).

39. Wang, S. *et al.* Noise-Induced Barren Plateaus in Variational Quantum Algorithms. *Nat Commun* **12,** 6961. ISSN: 2041-1723 (1 Nov. 2021).

40. Larocca, M. *et al. A Review of Barren Plateaus in Variational Quantum Computing* arXiv: 2405.00781 [quant-ph, stat]. http://arxiv.org/abs/2405.00781 (2024). Pre-published.

41. Arrasmith, A., Cerezo, M., Czarnik, P., Cincio, L. & Coles, P. J. Effect of barren plateaus on gradient-free optimization. *Quantum* **5,** 558. ISSN: 2521-327X (Oct. 2021).

42. Cai, Z. *et al. Quantum Error Mitigation* arXiv: 2210.00921 [quant-ph]. http://arxiv.org/abs/2210.00921 (2022). Pre-published.

43. McClean, J. R., Romero, J., Babbush, R. & Aspuru-Guzik, A. The Theory of Variational Hybrid Quantum-Classical Algorithms. *New J. Phys.* **18,** 023023. ISSN: 1367-2630 (Feb. 2016).

44. OMalley, P. J. J. *et al.* Scalable Quantum Simulation of Molecular Energies. *Phys. Rev. X* **6,** 031007 (July 2016).

45. Orus, R. Tensor Networks for Complex Quantum Systems. *Nat Rev Phys* **1,** 538–550. ISSN: 2522-5820 (Sept. 2019).

46. Bridgeman, J. C. & Chubb, C. T. Hand-Waving and Interpretive Dance: An Introductory Course on Tensor Networks. *J. Phys. A: Math. Theor.* **50,** 223001. ISSN: 1751-8121 (May 2017).

47. Baxter, R. J. Dimers on a rectangular lattice. *Journal of Mathematical Physics* **9,** 650–654 (1968).

48. Schollwoeck, U. The Density-Matrix Renormalization Group in the Age of Matrix Product States. *Annals of Physics* **326,** 96–192. ISSN: 00034916 (Jan. 2011).

49. Cirac, I., Perez-Garcia, D., Schuch, N. & Verstraete, F. Matrix Product States and Projected Entangled Pair States: Concepts, Symmetries, and Theorems. *Rev. Mod. Phys.* **93,** 045003. ISSN: 0034-6861, 1539-0756 (Dec. 2021).

50. Affleck, I., Kennedy, T., Lieb, E. H. & Tasaki, H. Rigorous results on valence-bond ground states in antiferromagnets. *Phys. Rev. Lett.* **59,** 799–802 (7 Aug. 1987).

51. Affleck, I., Kennedy, T., Lieb, E. H. & Tasaki, H. Valence bond ground states in isotropic quantum antiferromagnets. *Communications in Mathematical Physics* **115,** 477–528 (1988).

52. Dukelsky, J., Martín-Delgado, M. A., Nishino, T. & Sierra, G. Equivalence of the variational matrix product method and the density matrix renormalization group applied to spin chains. *Europhysics letters* **43,** 457 (1998).

53. Östlund, S. & Rommer, S. Thermodynamic Limit of Density Matrix Renormalization. *Phys. Rev. Lett.* **75,** 3537–3540 (19 Nov. 1995).

54. Tagliacozzo, L., de Oliveira, T. R., Iblisdir, S. & Latorre, J. I. Scaling of entanglement support for matrix product states. *Physical Review BCondensed Matter and Materials Physics* **78,** 024410 (2008).

55. Pollmann, F., Mukerjee, S., Turner, A. M. & Moore, J. E. Theory of finite-entanglement scaling at one-dimensional quantum critical points. *Physical review letters* **102,** 255701 (2009).

56. Zou, Y., Milsted, A. & Vidal, G. Conformal data and renormalization group flow in critical quantum spin chains using periodic uniform matrix product states. *Physical review letters* **121,** 230402 (2018).

57. Bañuls, M. C. Tensor Network Algorithms: A Route Map. *Annu. Rev. Condens. Matter Phys.* **14,** 173–191. ISSN: 1947-5454, 1947-5462 (Mar. 2023).

58. Swingle, B. Entanglement Renormalization and Holography. *Phys. Rev. D* **86,** 065007 (Sept. 2012).

59. Van de Wetering, J. *ZX-calculus for the Working Quantum Computer Scientist* arXiv: 2012.13966 [quant-ph]. http://arxiv.org/abs/2012.13966 (2023). Pre-published.

60. Stoudenmire, E. & Schwab, D. J. *Supervised Learning with Tensor Networks* in *Advances in Neural Information Processing Systems* **29** (Curran Associates, Inc., 2016).

61. Huggins, W., Patil, P., Mitchell, B., Whaley, K. B. & Stoudenmire, E. M. Towards Quantum Machine Learning with Tensor Networks. *Quantum Sci. Technol.* **4,** 024001. ISSN: 2058-9565 (Jan. 2019).

62. Gourianov, N. *et al.* A Quantum-Inspired Approach to Exploit Turbulence Structures. *Nat Comput Sci* **2,** 30–37. ISSN: 2662-8457 (1 Jan. 2022).

63. Lin, S.-H., Dilip, R., Green, A. G., Smith, A. & Pollmann, F. Real- and Imaginary-Time Evolution with Compressed Quantum Circuits. *PRX Quantum* **2,** 010342 (Mar. 2021).

64. Ran, S.-J. Encoding of Matrix Product States into Quantum Circuits of One- and Two-Qubit Gates. *Phys. Rev. A* **101,** 032310. ISSN: 2469-9926, 2469-9934 (Mar. 2020).

65. Schön, C., Solano, E., Verstraete, F., Cirac, J. I. & Wolf, M. M. Sequential Generation of Entangled Multiqubit States. *Phys. Rev. Lett.* **95,** 110503 (Sept. 2005).

66. Schoen, C., Hammerer, K., Wolf, M. M., Cirac, J. I. & Solano, E. Sequential Generation of Matrix-Product States in Cavity QED. *Phys. Rev. A* **75,** 032311. ISSN: 1050-2947, 1094-1622 (Mar. 2007).

67. Bañuls, M. C., Pérez-García, D., Wolf, M. M., Verstraete, F. & Cirac, J. I. Sequentially Generated States for the Study of Two-Dimensional Systems. *Phys. Rev. A* **77,** 052306 (May 2008).

68. Smith, A., Jobst, B., Green, A. G. & Pollmann, F. Crossing a Topological Phase Transition with a Quantum Computer. *Phys. Rev. Research* **4,** L022020. ISSN: 2643-1564 (Apr. 2022).

69. Chertkov, E. *et al.* Holographic Dynamics Simulations with a Trapped-Ion Quantum Computer. *Nat. Phys.* **18,** 1074–1079. ISSN: 1745-2481 (9 Sept. 2022).

70. Orús, R. A Practical Introduction to Tensor Networks: Matrix Product States and Projected Entangled Pair States. *Annals of Physics* **349,** 117–158. ISSN: 0003-4916 (Oct. 2014).

71. Evenbly, G. A Practical Guide to the Numerical Implementation of Tensor Networks I: Contractions, Decompositions, and Gauge Freedom. *Frontiers in Applied Mathematics and Statistics* **8.** ISSN: 2297-4687 (2022).

72. Vidal, G. Efficient Classical Simulation of Slightly Entangled Quantum Computations. *Phys. Rev. Lett.* **91,** 147902 (14 Oct. 2003).

73. Perez-Garcia, D., Verstraete, F., Wolf, M. M. & Cirac, J. I. Matrix product state representations. *arXiv preprint quant-ph/0608197* (2006).

74. Verstraete, F. & Cirac, J. I. Matrix product states represent ground states faithfully. *Physical Review B* **73.** ISSN: 1550-235X (Mar. 2006).

75. Hastings, M. B. An area law for one-dimensional quantum systems. *Journal of Statistical Mechanics: Theory and Experiment* **2007,** P08024–P08024. ISSN: 1742-5468 (Aug. 2007).

76. Tucci, R. R. *An Introduction to Cartan's KAK Decomposition for QC Programmers* arXiv: quant-ph/0507171. http://arxiv.org/abs/quant-ph/0507171 (2005). Pre-published.

77. Dborin, J. *et al.* Simulating Groundstate and Dynamical Quantum Phase Transitions on a Superconducting Quantum Computer. *Nat Commun* **13,** 5977. ISSN: 2041-1723 (1 Oct. 2022).

78. Zeng, B., Chen, X., Zhou, D.-L., Wen, X.-G., *et al. Quantum information meets quantum matter* (Springer, 2019).

79. Hastings, M. B. Solving Gapped Hamiltonians Locally. *Phys. Rev. B* **73,** 085115 (Feb. 2006).

80. Vanderstraeten, L., Haegeman, J. & Verstraete, F. Tangent-Space Methods for Uniform Matrix Product States. *SciPost Phys. Lect. Notes,* 7. ISSN: 2590-1990 (Jan. 2019).

81. Garcia-Escartin, J. C. & Chamorro-Posada, P. The SWAP Test and the Hong-Ou-Mandel Effect Are Equivalent. *Phys. Rev. A* **87,** 052330. ISSN: 1050-2947, 1094-1622 (May 2013).

82. Pfeuty, P. The One-Dimensional Ising Model with a Transverse Field. *Annals of Physics* **57,** 79–90. ISSN: 0003-4916 (Mar. 1970).

83. Nielsen, M. A. & Chuang, I. L. *Quantum Computation and Quantum Information* 10th anniversary ed. 676 pp. ISBN: 978-1-107-00217-3 (Cambridge University Press, Cambridge ; New York, 2010).

84. Barratt, F. *et al.* Parallel Quantum Simulation of Large Systems on Small NISQ Computers. *npj Quantum Inf* **7,** 1–7. ISSN: 2056-6387 (1 May 2021).

85. Vidal, G. Efficient Simulation of One-Dimensional Quantum Many-Body Systems. *Phys. Rev. Lett.* **93,** 040502 (4 July 2004).

86. Rommer, S. & Östlund, S. Class of ansatz wave functions for one-dimensional spin systems and their relation to the density matrix renormalization group. *Phys. Rev. B* **55,** 2164–2181 (4 Jan. 1997).

87. Haegeman, J. *et al.* Time-Dependent Variational Principle for Quantum Lattices. *Phys. Rev. Lett.* **107,** 070601 (Aug. 2011).

88. Haegeman, J., Lubich, C., Oseledets, I., Vandereycken, B. & Verstraete, F. Unifying Time Evolution and Optimization with Matrix Product States. *Phys. Rev. B* **94,** 165116. ISSN: 2469-9950, 2469-9969 (Oct. 2016).

89. Paeckel, S. *et al.* Time-Evolution Methods for Matrix-Product States. *Annals of Physics* **411,** 167998. ISSN: 00034916 (Dec. 2019).

90. Heyl, M. Dynamical Quantum Phase Transitions: A Review. *Rep. Prog. Phys.* **81,** 054001. ISSN: 0034-4885, 1361-6633 (May 2018).

91. Vajna, S. & Dóra, B. Topological Classification of Dynamical Phase Transitions. *Phys. Rev. B* **91,** 155127. ISSN: 1098-0121, 1550-235X (Apr. 2015).

92. Heyl, M. Scaling and Universality at Dynamical Quantum Phase Transitions. *Phys. Rev. Lett.* **115,** 140602. ISSN: 0031-9007, 1079-7114 (Oct. 2015).

93. Heyl, M., Polkovnikov, A. & Kehrein, S. Dynamical Quantum Phase Transitions in the Transverse-Field Ising Model. *Phys. Rev. Lett.* **110,** 135704. ISSN: 0031-9007, 1079-7114 (Mar. 2013).

94. Bonet-Monroig, X. *et al.* Performance Comparison of Optimization Methods on Variational Quantum Algorithms. *Phys. Rev. A* **107,** 032407. ISSN: 2469-9926, 2469-9934 (Mar. 2023).

95. Wecker, D., Hastings, M. B. & Troyer, M. Progress towards Practical Quantum Variational Algorithms. *Phys. Rev. A* **92,** 042303. ISSN: 1050-2947, 1094-1622 (Oct. 2015).

96. Pino, J. M. *et al.* Demonstration of the Trapped-Ion Quantum CCD Computer Architecture. *Nature* **592,** 209–213. ISSN: 1476-4687 (Apr. 2021).

97. D'Alessio, L., Kafri, Y., Polkovnikov, A. & Rigol, M. From quantum chaos and eigenstate thermalization to statistical mechanics and thermodynamics. *Advances in Physics* **65,** 239–362 (2016).

98. Jung, J.-H. & Noh, J. D. Guide to Exact Diagonalization Study of Quantum Thermalization. *Journal of the Korean Physical Society* **76,** 670–683. ISSN: 1976-8524 (Apr. 2020).

99. Klein Kvorning, T., Herviou, L. & Bardarson, J. H. Time-Evolution of Local Information: Thermalization Dynamics of Local Observables. *SciPost Physics* **13,** 080. ISSN: 2542-4653 (Oct. 2022).

100. Alhambra, Á. M. & Cirac, J. I. Locally Accurate Tensor Networks for Thermal States and Time Evolution. *PRX Quantum* **2,** 040331. ISSN: 2691-3399 (Nov. 2021).

101. White, C. D., Zaletel, M., Mong, R. S. K. & Refael, G. Quantum Dynamics of Thermalizing Systems. *Phys. Rev. B* **97,** 035127. ISSN: 2469-9950, 2469-9969 (Jan. 2018).

102. Rakovszky, T., Von Keyserlingk, C. & Pollmann, F. Dissipation-assisted operator evolution method for capturing hydrodynamic transport. *Physical Review B* **105,** 075131 (2022).

103. Osborne, T. J. Efficient Approximation of the Dynamics of One-Dimensional Quantum Spin Systems. *Phys. Rev. Lett.* **97,** 157202. ISSN: 0031-9007, 1079-7114 (Oct. 2006).

104. Kuwahara, T., Alhambra, Á. M. & Anshu, A. Improved Thermal Area Law and Quasilinear Time Algorithm for Quantum Gibbs States. *Phys. Rev. X* **11,** 011047. ISSN: 2160-3308 (Mar. 2021).

105. Bluhm, A., Capel, Á. & Pérez-Hernández, A. Exponential Decay of Mutual Information for Gibbs States of Local Hamiltonians. *Quantum* **6,** 650. ISSN: 2521-327X (Feb. 2022).

106. Huang, Y. Locally Accurate Matrix Product Approximation to Thermal States. *Science Bulletin* **66,** 2456–2457. ISSN: 20959273 (Dec. 2021).

107. Hauru, M., Van Damme, M. & Haegeman, J. Riemannian Optimization of Isometric Tensor Networks. *SciPost Phys.* **10,** 040. ISSN: 2542-4653 (Feb. 2021).

108. Hallam, A., Morley, J. & Green, A. G. The Lyapunov Spectrum of Quantum Thermalisation. *Nat Commun* **10,** 2708. ISSN: 2041-1723 (June 2019).

109. Dborin, J., Barratt, F., Wimalaweera, V., Wright, L. & Green, A. G. Matrix Product State Pre-Training for Quantum Machine Learning. *Quantum Sci. Technol.* **7,** 035014. ISSN: 2058-9565 (May 2022).

110. Schuld, M. & Petruccione, F. *Machine Learning with Quantum Computers* (Springer International Publishing, Cham, 2021).

111. Lloyd, S., Schuld, M., Ijaz, A., Izaac, J. & Killoran, N. Quantum Embeddings for Machine Learning. arXiv: 2001.03622 [quant-ph] (Feb. 2020). Pre-published.

112. Ventura, D. & Martinez, T. Quantum associative memory. *Information sciences* **124,** 273–296 (2000).

113. Phalak, K., Chatterjee, A. & Ghosh, S. Quantum random access memory for dummies. *Sensors* **23,** 7462 (2023).

114. Anschuetz, E. R. & Kiani, B. T. Quantum Variational Algorithms Are Swamped with Traps. *Nat Commun* **13,** 7760. ISSN: 2041-1723 (Dec. 2022).

115. Anschuetz, E. R. Critical Points in Quantum Generative Models. arXiv: 2109.06957 [quant-ph] (Jan. 2023). Pre-published.

116. Sim, S., Johnson, P. D. & Aspuru-Guzik, A. Expressibility and Entangling Capability of Parameterized Quantum Circuits for Hybrid Quantum-Classical Algorithms. *Advanced Quantum Technologies* **2,** 1900070. ISSN: 2511-9044 (2019).

117. Akshay, V., Philathong, H., Morales, M. E. S. & Biamonte, J. D. Reachability Deficits in Quantum Approximate Optimization. *Phys. Rev. Lett.* **124,** 090504 (Mar. 2020).

118. McClean, J. R., Boixo, S., Smelyanskiy, V. N., Babbush, R. & Neven, H. Barren Plateaus in Quantum Neural Network Training Landscapes. *Nat Commun* **9,** 4812. ISSN: 2041-1723 (1 Nov. 2018).

119. Rudolph, M. S. *et al.* Trainability barriers and opportunities in quantum generative modeling. *npj Quantum Information* **10,** 116 (2024).

120. Cerezo, M., Sone, A., Volkoff, T., Cincio, L. & Coles, P. J. Cost Function Dependent Barren Plateaus in Shallow Parametrized Quantum Circuits. *Nat Commun* **12,** 1791. ISSN: 2041-1723 (Dec. 2021).

121. Verdon, G. *et al. Learning to Learn with Quantum Neural Networks via Classical Neural Networks* arXiv: 1907.05415 [quant-ph]. http://arxiv.org/abs/1907.05415 (2022). Pre-published.

122. Pesah, A. *et al.* Absence of Barren Plateaus in Quantum Convolutional Neural Networks. *Phys. Rev. X* **11,** 041011. ISSN: 2160-3308 (Oct. 2021).

123. Arrasmith, A., Holmes, Z., Cerezo, M. & Coles, P. J. Equivalence of Quantum Barren Plateaus to Cost Concentration and Narrow Gorges. *Quantum Sci. Technol.* **7,** 045015. ISSN: 2058-9565 (Aug. 2022).

124. Grant, E., Wossnig, L., Ostaszewski, M. & Benedetti, M. An Initialization Strategy for Addressing Barren Plateaus in Parametrized Quantum Circuits. *Quantum* **3,** 214. ISSN: 2521-327X (Dec. 2019).

125. Skolik, A., McClean, J. R., Mohseni, M., van der Smagt, P. & Leib, M. Layerwise Learning for Quantum Neural Networks. *Quantum Mach. Intell.* **3,** 5. ISSN: 2524-4906, 2524-4914 (June 2021).

126. Rieser, H.-M., Köster, F. & Raulf, A. P. Tensor Networks for Quantum Machine Learning. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* **479,** 20230218 (July 2023).

127. Dborin, J., Barratt, F., Wimalaweera, V., Wright, L. & Green, A. G. Matrix Product State Pre-Training for Quantum Machine Learning. *Quantum Sci. Technol.* **7,** 035014. ISSN: 2058-9565 (May 2022).

128. Peruzzo, A. *et al.* A variational eigenvalue solver on a photonic quantum processor. *Nature Communications* **5.** ISSN: 2041-1723 (July 2014).

129. Farhi, E., Goldstone, J. & Gutmann, S. A Quantum Approximate Optimization Algorithm. arXiv: 1411.4028 [quant-ph] (2014).

130. Xiao, H., Rasul, K. & Vollgraf, R. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747* (2017).

131. Friedrich, L. & Maziero, J. Avoiding Barren Plateaus with Classical Deep Neural Networks. *Phys. Rev. A* **106,** 042433 (Oct. 2022).

132. Grimsley, H. R., Economou, S. E., Barnes, E. & Mayhall, N. J. An Adaptive Variational Algorithm for Exact Molecular Simulations on a Quantum Computer. *Nat Commun* **10,** 3007. ISSN: 2041-1723 (July 2019).

133. Galda, A., Liu, X., Lykov, D., Alexeev, Y. & Safro, I. *Transferability of Optimal QAOA Parameters between Random Graphs* in *2021 IEEE International Conference on Quantum Computing and Engineering (QCE)* 2021 IEEE International Conference on Quantum Computing and Engineering (QCE) (Oct. 2021), 171–180.

134. Cheng, S., Wang, L. & Zhang, P. Supervised Learning with Projected Entangled Pair States. *Phys. Rev. B* **103,** 125117. ISSN: 2469-9950, 2469-9969 (Mar. 2021).

135. Kong, F., Liu, X.-y. & Henao, R. Quantum Tensor Network in Machine Learning: An Application to Tiny Object Classification. arXiv: 2101.03154 [cs] (Jan. 2021). Pre-published.

136. Opitz, D. & Maclin, R. Popular ensemble methods: An empirical study. *Journal of artificial intelligence research* **11,** 169–198 (1999).

137. Wright, L. *et al.* Deterministic Tensor Network Classifiers. arXiv: 2205.09768 [cond-mat, physics:quant-ph] (May 2022). Pre-published.

138. Manual, A. B. An introduction to statistical learning with applications in R (2013).

139. Chen, J., Stoudenmire, E. & White, S. R. Quantum Fourier Transform Has Small Entanglement. *PRX Quantum* **4,** 040318 (4 Oct. 2023).