Distributed Matrix Product State Simulations of Large-Scale Quantum Circuits

Aidan Dang

A thesis presented for the degree of

Master of Science

under the supervision of

Dr. Charles Hill and Prof. Lloyd Hollenberg

School of Physics
The University of Melbourne

20 October 2017

Abstract

Before large-scale, robust quantum computers are developed, it is valuable to be able to classically simulate quantum algorithms to study their properties. To do so, we developed a numerical library for simulating quantum circuits via the matrix product state formalism on distributed memory architectures. By examining the multipartite entanglement present across Shor's algorithm, we were able to effectively map a high-level circuit of Shor's algorithm to the one-dimensional structure of a matrix product state, enabling us to perform a simulation of a specific 60 qubit instance in approximately 14 TB of memory: potentially the largest non-trivial quantum circuit simulation ever performed. We then applied matrix product state and matrix product density operator techniques to simulating one-dimensional circuits from Google's quantum supremacy problem with errors and found it mostly resistant to our methods.

Declaration

I declare the following as original work:

- In chapter 2, the theoretical background described up to but not including section 2.5 had been established in the referenced texts prior to this thesis. The main contribution to the review in these sections is to provide consistency amongst competing conventions and document the capabilities of the numerical library we developed in this thesis. The rest of chapter 2 is original unless otherwise noted.
- The techniques and experimental results of chapters 3 and 4 are original unless otherwise stated.

I declare that the code for our numerical library was written solely by the author, where any external libraries called have otherwise been referenced in this text.

Aidan Dang
October 2017

Acknowledgements

Most importantly, I give my gratitude to Doctor Charles Hill and Professor Lloyd Hollenberg for their guidance and vision in directing this work, for providing me with the resources necessary to produce the results of this thesis and for pushing me to realise my potential.

In aiding the development of our numerical library and the production of our results, I acknowledge that this work was supported by resources provided by the Pawsey Supercomputing Centre with funding from the Australian Government and the Government of Western Australia.

To The University of Melbourne, and especially the School of Physics, I am thankful to have been in an environment that would fuel the growth of my abilities. In order that I may focus on my studies, I would also like to thank Ms Rose Cooney, Ms Jenny de Boer and Associate Professor Michelle Livett for greatly simplifying the daunting administrative matters that I was faced with.

Finally, to all the friends, classmates, teachers and students that I have ever had, and anyone else I have ever learned something from, I hope that you might have gained something from me in return.

Contents

Chapter 1

Introduction

Despite the vast progress made in order to realise modern computing devices from a mathematical model, the Turing machine [1], this underlying model imposes limitations on the types of problems that such a 'classical' computer may solve, and the efficiency with which it might do so. Although the time required to solve a computable problem on a classical machine does decrease if we use more or faster hardware, there exist particularly consequential problems that, given a sufficiently large instance, may not be solved with any reasonable amount of time or hardware using our best known classical algorithms.

This presents several paths forward. Lacking proof that our current classical algorithms for these problems obtain solutions with optimal efficiency, one approach might be to develop more efficient algorithms for our existing devices, an active area of research in computer science today. Another is to subscribe to a different model of computation, in which algorithms exist for solving these problems efficiently. Indeed, with a *quantum* computer based on the quantum circuit model of computation, solutions for problems such as integer factorisation [2] may be obtained efficiently.

The concept of these quantum machines that potentially possess some inherent speed advantage over their classical counterparts motivates their physical construction, with the experimentation and engineering required in their implementation needing to stave off the effects of noise and decoherence in order to scale. Before such large-scale quantum computers can be built, methods to simulate their specific behaviours and interactions with quantum circuits should still prove valuable. Though the ability to perform such simulations would still be subject to the limitations of classical computation, it may, for example, allow us to study a quantum circuit's resilience toward these noise effects on specific proposed hardware implementations.

Despite this quantum advantage over currently known classical algorithms for these problems, the presence of an innate advantage is still uncertain. However, in order to highlight perceived speed differences between the quantum and classical models, several 'quantum supremacy' problems have been proposed, with the aim of being efficiently solvable on quantum hardware but not efficiently solvable classically. Often, these problems claim a 'supremacy point', a sufficiently large instance size that a reasonable amount of classical hardware may not solve in a reasonable amount of time; a quantum device that successfully solves problems of this size are then said to demonstrate quantum supremacy. Currently, physical implementations of quantum computers are unable to claim quantum supremacy.

Quantum computation and algorithms

If the states that comprise a quantum mechanical system form a d-dimensional Hilbert space $\mathcal{H}$, then n such systems form the d^n-dimensional Hilbert space $\mathcal{H}^{\otimes n}$. If we consider a system of 2-dimensional subsystems, then the values 0 and 1 can be assigned to these levels to produce a

quantum bit or *qubit*, the quantum counterpart to the bit in classical computation. Since a state of this composite system resides as an element within $\mathcal{H}^{\otimes n}$, familiar quantum properties such as superposition and entanglement are applicable. For example, a collection of n bits allows us to store exactly one value out of the 2^n possible binarily representable values, whilst a set of n quantum bits may describe a superposition of all these 2^n values. It would then appear that any speed advantage of a quantum computer arises from these exclusively quantum properties. However, the extent to which each of these properties contributes to the perceived advantage is not fully understood [3].

Just as classical bits are manipulated as electrical signals passing through circuits of logic gates in a classical computer, the quantum circuit model of computation involves acting quantum gates on one or more of the quantum system's constituent subsystems. In quantum mechanical terms, these gates act as unitary transformations on the quantum state. Therefore, unlike most classical logic gates, applying a quantum gate is a reversible operation, which has consequences on the information entropy content of a state [4]. Furthermore, measurement of a quantum state to extract information from it, such as the results of a quantum algorithm, necessarily alters the state via a collapse of superposition and entanglement, unlike in a system of classical bits. Quantum algorithmic design must therefore keep these constraints in mind.

It is also worth mentioning that other quantum models of computation exist. Notably, the quantum Turing machine as a generalisation of the classical Turing machine is considered computationally equivalent to the quantum circuit model [5], and implementations based on these models are referred to as universal quantum computers. Additionally, an adiabatic quantum computer [6] functions by initialising a system to the ground state of a simple Hamiltonian and adiabatically evolving it so that the system remains in the ground state, but of a Hamiltonian where this ground state encodes the solution to the relevant problem.

The efficiency of any computational algorithm often refers to its time complexity: the scaling of the algorithm's running time with respect to its input size. Specifically, we express an algorithm's time complexity in terms of its time usage's asymptotic behaviour using Bachmann–Landau notation. To demonstrate, classical lookup of a single item from a black-box database of N items has an average time complexity of $\mathcal{O}(N)$, as each item must be sequentially queried until the relevant entry is found. However, Grover's algorithm [7], performed on a quantum computer, allows this task to be performed in $\mathcal{O}(\sqrt{N})$ time, resulting in a quadratic advantage over the classical approach. When a symmetric-key cryptographic algorithm such as AES [8] with a key of length n bits is used in place of this black-box database, the $\mathcal{O}(N = 2^n)$ lookups for the classical approach is reduced to $\mathcal{O}(\sqrt{N} = 2^{n/2})$ operations through Grover's algorithm, and therefore halving the effective key length.

An algorithm is said to be *efficient* if its time complexity is bounded above by some polynomial. With this definition, Grover's algorithm is not sufficient to efficiently break AES encryption by searching through all possible keys. Factorisation of an integer of length n digits is one example of a problem that can be solved in polynomial time with respect to n on a quantum computer through Shor's algorithm [2], but lacks any known algorithm to do so efficiently on a classical machine. Therefore, quantum computers currently exhibit an exponential speed advantage over classical computers in integer factorisation. Many of the public-key cryptographic protocols [9] presently used for online communications are predicated on the difficulty of integer factorisation and the discrete logarithm problem (which Shor's algorithm also solves efficiently), so post-quantum cryptographic protocols resistant to attacks by quantum computers have been proposed [10] in response.

Supremacy problems

Given the multitude of quantum algorithms that provide a computational speedup over their fastest known classical counterparts [11], it is still an open question in computer science as to whether or not these quantum computers, based on the various quantum models of computation such as the circuit model, innately possess some computational advantage over our classical devices built upon the Turing machine. The difficulty in proving the existence of this advantage, i.e. the existence of quantum supremacy, stems from bounding the computational power of classical machines; though some classical algorithms are bested by quantum algorithms in terms of their time complexity, these quantum algorithms must demonstrate their advantage over all possible classical algorithms in order to prove quantum supremacy, and not just over the classical algorithms that are known.

Even if quantum supremacy were theoretically possible, there still remains the experimental challenges in building a physical machine to demonstrate it. As a physical quantum device interacts with its environment over the course of a circuit, effects such as quantum decoherence and noise begin to alter its internal quantum state in ways extraneous to the circuit's gates, potentially leading to errors in any results obtained through measurement. Quantum error correction codes such as the surface code [12] have been developed to increase the resilience of physical quantum computers toward noise and decoherence, at the cost of requiring greater redundancy in the number of qubits.

Despite the theoretical and experimental challenges in demonstrating quantum supremacy, problems designed to be efficiently solvable through a quantum model of computation but not through any known classical algorithm have been proposed as tests [13]. Often, these are sampling problems: a quantum device will perform some efficient manipulation of a quantum state and sample measurements from the resulting underlying probability distribution. It is then argued that classical computers cannot efficiently reproduce this probability distribution to sample from, given some specification of the quantum operation involved (e.g. the quantum circuit). In the case of boson sampling [14], the device is a linear optical quantum computer that manipulates Fock states of input photons via a linear optical network. Given the best known classical algorithms for simulating the results of these networks [14], it is estimated that instances of boson sampling with 50 photons exceed the capabilities of modern classical hardware [15].

In the quantum circuit model, IQP (instantaneous quantum polynomial-time) [16] circuits and the circuits proposed by Google's Boixo et al. [17] involve sampling from the measurement results of circuits randomly generated according to a set of rules. In both examples, noise prevents a physical quantum computer from exactly sampling from the ideal distribution. In [17], a measure of how well a noisy quantum computer may sample from the ideal distribution produced from one of their circuits is related to the effective per-gate error rate of the physical implementation. This allows a physical quantum computer's effectiveness at Google's sampling problem to be compared to a classical simulation's, despite being subject to the effects of noise and errors.

Simulation techniques

For a two-level quantum system of Hilbert space $\mathcal{H}$ with basis elements labelled $\{|0\rangle, |1\rangle\}$ used to represent a qubit, the evolution of a quantum system of n qubits in state $|\psi\rangle$ through the unitary transformations of a quantum gate takes place within the 2^n-dimensional Hilbert space $\mathcal{H}^{\otimes n}$, whose basis elements are

$$\{|0\ldots0\rangle, |0\ldots1\rangle, \ldots, |1\ldots1\rangle\} \equiv \{|0\rangle \otimes \ldots \otimes |0\rangle, |0\rangle \otimes \ldots \otimes |1\rangle, \ldots, |1\rangle \otimes \ldots \otimes |1\rangle\}.$$

Computationally, $|\psi\rangle$ can be described as a vector storing its 2^n complex coefficients in this basis for its Hilbert space $\mathcal{H}^{\otimes n}$, and linear operators such as the unitary gate transformations may be applied via matrix multiplication with its matrix representation in the same basis. Software packages [18, 19] that perform simulations of the evolution of quantum states in this manner are accordingly referred to as state vector simulators. Despite any properties of the quantum state that might change over the course of, say, a quantum circuit, the state vector formalism imposes a strict $\mathcal{O}(2^n)$ space complexity to simply store this state vector. For example, the 2 PiB required to store an $n = 48$ qubit state vector with 64 bit per complex coefficient approaches the limit of memory available to modern supercomputers [15]. Furthermore, simulating quantum state evolution via matrix multiplication on these state vectors has an exponentially scaling time cost. Therefore, state vector simulations cannot currently be performed efficiently on classical hardware.

An alternative representation of a quantum state $|\psi\rangle$ might instead adaptively scale its time and space requirements according to some physical property of the state. Based on the idea of tensor networks [20], Vidal [21] proposed the idea of using matrix product states (MPSs) as a representation of quantum states for simulating quantum circuits. The MPS formalism was originally used for other applications in condensed matter physics, namely for determining ground states of quantum systems through the density matrix renormalisation group (DMRG) algorithm [22] and simulating the time evolution of quantum states through a given Hamiltonian [23]. In the MPS representation of a quantum state, if subsystems are laid out in a one-dimensional lattice, then only information about a subsystem and its entanglement with the adjacent subsystems needs to be stored. Therefore, the space and time requirements to store and operate on an MPS may scale with the amount of entanglement present in the represented quantum state, where the 'amount' of entanglement is quantified by a particular measure. For Hamiltonians, quantum circuits or other quantum operators that do not introduce large amounts of entanglement, the MPS representation may be used to efficiently simulate [23] these quantum systems on classical hardware.

Though the state vector and matrix product state formalisms are suited for simulation of pure quantum states, modelling errors in a simulation of physical quantum hardware invites consideration of the density operator $\rho = \sum_i |\psi_i\rangle\langle\psi_i|$ for mixed states. The density matrix formalism for representing the density operator of a d-dimensional state involves storing a matrix of dimension $d \times d$, and is the mixed state counterpart for the state vector formalism. Similarly, the matrix product state formalism may be generalised to a matrix product density operator formalism that also scales its time and space requirements for the amount of entanglement.

Outline

This thesis is focused on the development and features of our distributed-memory matrix product state software library, QCMPS, as well as a demonstration of its capabilities by simulating Shor's algorithm and the Google supremacy problem. Chapter 2 provides a more formal introduction to the tensor network formalism that underpins the mechanics of MPSs, and introduces the functionality of QCMPS with a comparison to existing quantum simulators. In chapter 3, we benchmark QCMPS with a high-level simulation of Shor's algorithm following the method of [24] with additional optimisations to better suit the computation to MPSs. Using approximately 14 TB of memory on the Magnus supercomputer [25], we were able to simulate an instance of Shor's algorithm on 60 qubits, being potentially the largest high-level but non-trivial quantum circuit ever simulated. Finally, we investigate the one-dimensional Google supremacy problem with and without errors to determine if any of our MPS techniques might serve to refine the estimate for its supremacy point.

Chapter 2

QCMPS Library Showcase

The matrix product state formalism is a method for representing quantum states alternative to the more familiar state vector formalism. As we discuss in this chapter, the time and space requirements for storing and manipulating these matrix product states (MPSs) scale with the amount of entanglement within the quantum state, making it a viable alternative for use in classical simulations of certain quantum systems [21] when compared to the state vector formalism with its more rigid requirements.

Since the use of MPSs for classical simulation of quantum circuits forms the basis of this thesis, we will provide an introduction to their mechanics and an overview of the operations that can be performed with them. To begin, we introduce the use of tensor networks [20] for its convenient pictorial representations of the methods performed. The mechanics of MPSs can then be built on this foundation.

To implement these operations, we have developed the numerical library *QCMPS* for simulating quantum circuits. It is specifically tailored for this purpose, as opposed to simulating other systems relevant to condensed matter physics [22, 23], so we document our choice of conventions and our design decisions here.

2.1 Introduction to tensor networks

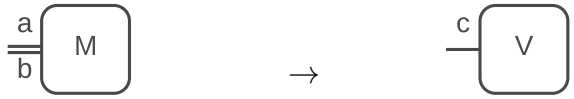
For this thesis, we define a tensor T of rank n to be a simple n-dimensional array of scalars, similar to [26]. We may index individual elements of these tensors through indices where, for us, raised or lowered indices are not relevant. Each index has some associated dimension that we denote with the upper case of the index (e.g. M_{ab} is an element of a rank-2 tensor, or matrix, of size $A \times B$). Following [20], we can represent such rank-n tensors diagrammatically as shapes with n emerging lines, where each line is associated with a single dimension of the tensor. Examples of low-rank tensors are shown in Fig. 2.1. Through these diagrams, we can examine some basic tensor operations.

(a) Scalar S (b) Vector V of elements V_a (c) Matrix M of elements M_{ab}

Figure 2.1: Pictorial representations of low-rank tensors

2.1.1 Reshape and transpose

Reshaping a tensor involves redistributing its elements over a different set of indices. For example, vectorising a matrix M into vector V yields $V_c = M_{ab}$ as shown in Fig. 2.2. Note that a tensor's size is equal to the product of its indices' dimensions, and the reshape operation must conserve this quantity. This also allows us to insert and remove indices of unit dimension as we please: [26] refers to these as expanding and squeezing respectively. As such, the reshape operation may alter the number of emerging lines from a tensor's diagrammatic representation, as well as the corresponding dimensions. It is also worth mentioning that, computationally, a tensor must be vectorised in order to be stored linearly in memory. The two storage schemes, row-major and column-major storage, produce different results when a tensor is reshaped, so it is important to be aware of a numerical library's convention.



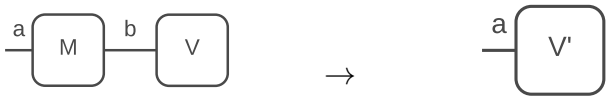
(a) Matrix before reshape (b) Vector after reshape

Figure 2.2: Matrix to vector reshape example

A generalised tensor transpose is a reordering of a tensor's indices. For the familiar matrix transpose, we have $M^{\top}_{ba} = M_{ab}$, and for a rank-4 tensor, we might have $T'_{dbac} = T_{abcd}$. Thus, a transpose operation does not alter a tensor's diagrammatic representation, but is a useful complement to the reshape operation. Composing reshapes and transposes allows us to manipulate the layout of a tensor.

2.1.2 Contraction and tensor product

An Einstein summation notation expression for a tensor contraction can be represented by joining lines corresponding to summation indices, creating a tensor network diagram. We allow an Einstein summation expression to repeat an index more than twice (since raised and lowered indices are not relevant to us), so a tensor network diagram of such an expression would join more than two index lines together. Only indices of equal dimension may be contracted. Fig. 2.3 shows a simple example of a tensor network.



(a) Tensor network of $M_{ab}V_b$ (b) Resulting contraction

Figure 2.3: Tensor network example of applying the linear transform $V'_a = M_{ab}V_b$

A tensor product T of two tensors A and B has the combined indices of A and B such that each element of T is the scalar product of the element in A and the element in B with the corresponding indices. For example, the outer product of two vectors yields $T_{ab} = A_a B_b$. Pictorially, this can be thought of as reshaping both A and B to add a new index of unit dimension and contracting over this new index. An addition example is provided in Fig. 2.4.

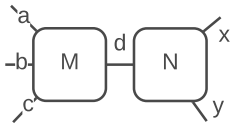


Figure 2.4: Tensor network example for the outer product $M_{abc}N_{xy}$. Note that M and N are expanded to produce contraction index d of unit dimension.

2.1.3 Decomposition

Contracting tensors over specific indices serves as a natural way to evaluate tensor network diagrams. However, for the purposes of this thesis, it is also important that we have some method of decomposing a tensor into a contraction of multiple simpler tensors in a tensor network diagram. First, we consider the decomposition of a rank-2 tensor, matrix M. There are several linear algebra routines for decomposing a matrix:

- The trivial decomposition

$$M_{ab} = \begin{cases} M_{ac}\delta_{cb}, & A \geq B \\ \delta_{ac}M_{cb}, & A < B \end{cases}, \tag{2.1}$$

  where δ is the Kronecker delta.

- The QR decomposition $M_{ab} = Q_{ac}R_{cb}$, where Q is a unitary matrix and R is an upper triangular matrix.

- The singular value decomposition (SVD)

$$M_{ab} = U_{ac}S_cV_{cb}, \tag{2.2}$$

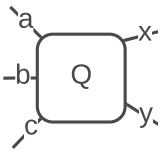
  where $C \leq \min(A, B)$, U and V are unitary matrices and S has real, non-negative elements. In tensor network diagrams, the vector S of singular values is often promoted to a diagonal, square matrix such that $M_{ab} = U_{ac}S_{cd}V_{db}$.

The SVD (and QR decomposition when used with column pivoting) is rank-revealing, so we can use these decompositions to minimise the number of elements stored in the resulting tensors. For the SVD, this means only storing elements corresponding to non-zero entries of S in Eq. (2.2).

Decomposing a general tensor may be done by transposing and reshaping it into a matrix so that a matrix decomposition such as the SVD can be used. Following the matrix decomposition, the resulting matrices can be reshaped and transposed to restore the indices of the original tensor. This process is demonstrated in Fig. 2.5, and is henceforth used when applying the SVD to a general tensor. Additional decompositions can be performed on these resulting tensors to further separate indices of the original tensor.

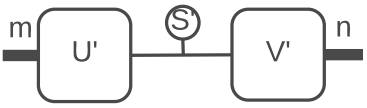
2.2 Quantum circuits as tensor networks

At its most basic level, classical computation relies on performing operations on bits, digital units of information whose state can be represented by either 0 or 1. In quantum computation, the equivalent unit of information is the quantum bit, or qubit, a two-level quantum system represented by a unit vector in $\mathbb{C}^2$ with the standard complex inner product and norm. As such, we can represent a qubit in our tensor network diagrams as a vector with index of dimension 2. In general, however, a d-level quantum system may underpin a qudit, which can then

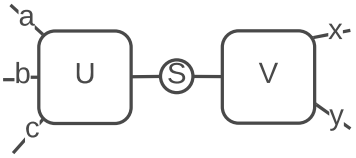


(a) Example tensor Q, for splitting $\{a, b, c\}$ from $\{x, y\}$

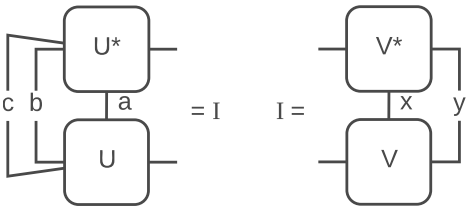
(b) Reshape of Q to Q'



(c) SVD of Q' per Eq. (2.2)



(d) Restoration of original indices and promotion of vector S' to square, diagonal matrix S



(e) Unitarity conditions on U and V. I here represents the appropriately sized identity matrix and the asterisk refers to an element-wise complex conjugation.

Figure 2.5: Decomposition of tensor Q to group indices $\{a, b, c\}$ and $\{x, y\}$ amongst each other

be represented by a unit vector in $\mathbb{C}^d$ or in a tensor network by a vector with an index of dimension d. These *state vectors* for a single qubit or qudit are conventionally written in terms of a complete, orthonormal basis. If the two basis elements of a qubit, for example, correspond to the classical states 0 and 1, then we have what we call the computational basis, in which

$$\begin{aligned} |\psi\rangle &= \alpha\,|0\rangle + \beta\,|1\rangle\,, \\ \langle\psi|\psi\rangle &= |\alpha|^2\,\langle 0|0\rangle + |\beta|^2\,\langle 1|1\rangle \\ &= |\alpha|^2 + |\beta|^2 \\ &= 1. \end{aligned}$$

for some arbitrary state $|\psi\rangle$ in our familiar bra-ket notation. Due to this normalisation, measurement of this qubit in the computational basis carries the interpretation that $|\psi\rangle$ collapses to $(\alpha/|\alpha|)\,|0\rangle$ with probability $|\alpha|^2$, or to $(\beta/|\beta|)\,|1\rangle$ with probability $|\beta|^2$. This readily extends to general qudits.

Analogous to logic gates for classical circuits, quantum circuits make use of quantum gates to manipulate state. Quantum gates acting on a single d-level qudit are unitary transforms, which can be written as $d \times d$ unitary matrices in the same orthonormal basis of the state vector it should act on. In a tensor network, these quantum gates are represented as a simple rank-2 tensor which contracts with the index of a qudit's state vector. One common example

of a quantum gate is the Hadamard gate for qubits, which performs the operation

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

As such, the Hadamard gate has a matrix representation

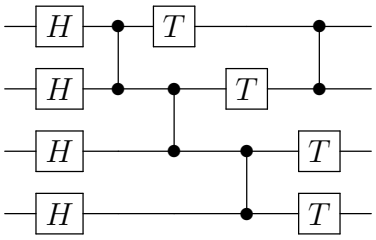
$$H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{2.3}$$

in the computational basis.

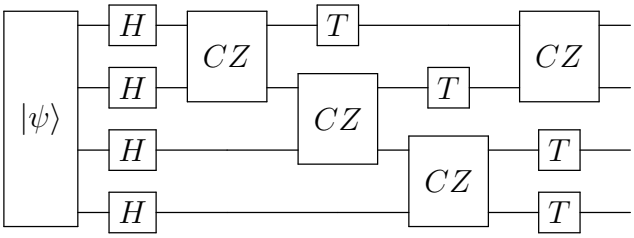
In a system of multiple qudits, the set of basis vectors of the overall state vector is the set of all possible tensor products of the constituent subsystems' basis vectors. Therefore, a tensor network diagram of, say, n qubits has n index lines, each of dimension 2. Single qudit gates can be applied as tensor contractions to a specific line corresponding to the qudit to operate on. Furthermore, we may also operate multi-qudit gates that contract to multiple qudit indices. An example circuit represented as a tensor network diagram in this manner is shown in Fig. 2.6b.

Since a state vector's tensor diagram can be reshaped to combine qudit indices, an n qubit tensor with n lines of dimension 2 storing 2^n elements is equivalent to a single 2^n level qudit, or any other combination of d_i level qudits resulting in a total dimensionality of 2^n. Similarly, multiple qudit gates can be reshaped to an equivalent single qudit unitary operation.

Quantum circuits are used to display the layout of quantum gates. Typically, they are drawn with horizontal lines representing qubits initialised to some indicated state (usually $|0\rangle$), so the initial state vector will be the tensor product of these subsystems. Circuits are then composed by placing gates on single lines for single qubit gates, or on multiple lines for multiple qubit gates. Thus, a quantum circuit is simply a tensor network diagram: we can interpret all gate operations as tensor contractions on a multiple-qubit state vector. An example circuit with its equivalent tensor network is provided in Fig. 2.6.



(a) Example circuit acting on state $|\psi\rangle$ of four qubits



(b) The corresponding tensor network. Note the multiple index lines originating from the state vector for $|\psi\rangle$, each representing a distinct qubit subsystem.

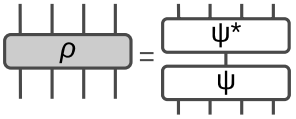
Figure 2.6: An example quantum circuit and its corresponding tensor network. For now, the explicit matrix representations of the as-yet unintroduced gates are not important.

As stated earlier, measurement of a single qudit in a particular basis collapses the state into a single basis state, with some probability related to the square of the absolute value of the state vector's corresponding coefficient. At this point, we introduce the pure state density matrix

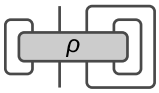
$$\rho = |\psi\rangle\langle\psi|\,.$$

In a tensor network, we construct this density matrix by taking the tensor product of a state vector with its complex conjugate. The diagonal elements of this density matrix now indicate the probability of measuring a particular state. The density matrix for a system of multiple qudits, by extension, is the tensor product of a multiple-qudit state vector with its complex conjugate, shown in Fig. 2.7a.

In a system of multiple qudits, measurement of a single qudit also requires the density matrix of the entire state vector. However, in order to arrive at a reduced density matrix for the qudit system to be measured, we now 'trace over' all other qudit subsystems by contracting pairs of index lines corresponding to each other qudit (so that the only free indices of the reduced density matrix correspond to the qudit of interest) as per Fig. 2.7b. The diagonal elements of this reduced density matrix correspond to the probabilities p_m of measuring value m from the qudit. When a particular value m is measured for the given qudit, all elements in the original state vector corresponding to m for that qudit are rescaled by $1/\sqrt{p_m}$, and all other elements are set to zero. In this way, we can also interpret measurements of single qubits in quantum circuit diagrams.



(a) Tensor network of the pure density matrix $\rho = |\psi\rangle\langle\psi|$, where the contracted index has unit dimension



(b) The reduced density matrix of the second qudit system.

Figure 2.7: Tensor networks for the density matrix and reduced density matrix

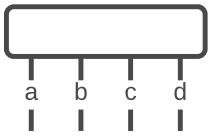
2.3 Matrix Product States

In contrast to a state vector simulation of a quantum circuit, which performs gate operations as tensor contractions on the entire state vector (of size 2^n for an n qubit system), the matrix product state (MPS) formalism was developed for its adaptive computational resource requirements, and also allows simulating gates as local operations. To accomplish this, the state vector tensor is sequentially decomposed so that each qudit line is joined to a distinct tensor which are laid out linearly, producing a tensor network like Fig. 2.8b. Therefore, each tensor in an MPS chain has three index lines: one representing the qudit's index in the original state vector, and index lines to the left and right contracting to adjacent qudits. Applying single qudit gates now only requires contraction with the relevant qudit tensors rather than the entire state vector. If the SVD, Eq. (2.2), is used as the matrix decomposition, we also have the vectors of singular values between each qudit tensor. We refer to these vectors as the bond tensors, and although they can be contracted into an adjacent qudit tensor, we choose to keep them separate. Why we choose to do so will be explained shortly.

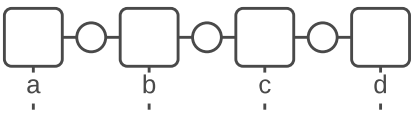
In the following section, we demonstrate the mechanics of the MPS formalism as operations performed by our numerical library QCMPS.

2.4 QCMPS

We have developed a C++ library titled QCMPS for applying the MPS formalism specifically to simulating quantum circuits. Due to the limited computational resources of a single classical computer, we require the use of the Message Passing Interface (MPI) [27] to make use of



(a) A state vector of four qudits



(b) A matrix product state representation for this system. The intermediate circles represent the vectors of singular values, or bond tensors, when the SVD is used.

Figure 2.8: Tensor networks for a quantum state represented in the state vector and matrix product state descriptions

distributed memory systems to increase the amount of memory and parallelism available to us. Typically, a library for dense tensor contraction might be built upon [28] or [29]. However, these libraries do not include support for any matrix decompositions, so they are unsuitable for a complete, distributed memory MPS library. Such routines are provided in ScaLAPACK [30], but reference implementations of ScaLAPACK only allow for indexing arrays through 32 bit integers, ultimately limiting the size of the tensors we can work with. Furthermore, SVD in ScaLAPACK is restricted to a slower algorithm based on QR iteration [31]. With these limitations in mind, QCMPS is built on Elemental [32], chosen for its faster divide and conquer SVD algorithm [33] and 64 bit indexing when used with a compatible Basic Linear Algebra Subprograms (BLAS) and Linear Algebra Package (LAPACK) [34] implementation, such as OpenBLAS [35] or the Intel Math Kernel Library.

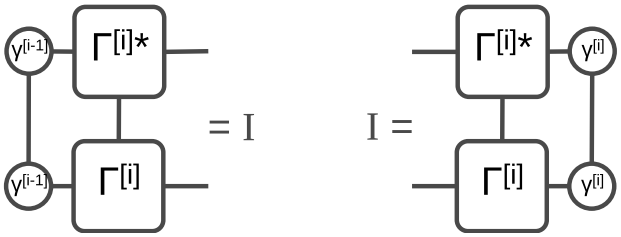
2.4.1 MPS canonical form

In section 2.2, we defined a method of making measurements of individual subsystems in the state vector formalism. This involved calculating the reduced density matrix of the quantum subsystem to be measured, say qudit q, through contractions on a tensor network. If the state vector is decomposed into an MPS, similar contractions are required: we still trace over all qudits other than q, producing a tensor network such as Fig. 2.9b. However, if each 'loop' left and right of q's tensors is equal to the identity matrix, the reduced density matrix can be calculated locally. These loops are defined in Fig. 2.9a. We define a qudit to be in 'left canonical' ('right canonical') form if it produces a left (right) loop resulting in the identity matrix. The reduced density matrix of q can be locally calculated if all qudits left (right) of q are in left (right) canonical form with a tensor network given by Fig. 2.9c. We define an MPS to be in canonical form if every qudit is in both left and right canonical forms. In this case, the density matrix of every qudit may be calculated locally, and each bond tensor $\gamma^{[m]}$ at bipartition m contains the Schmidt coefficients for m. Therefore, the bond tensors may be used to determine the entropy of entanglement

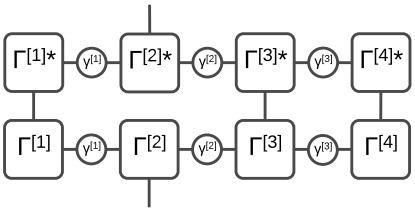
$$H(m) = -\sum_{\alpha_m} \left|\gamma^{[m]}_{\alpha_m}\right|^2 \log_2 \left|\gamma^{[m]}_{\alpha_m}\right|^2 \tag{2.4}$$

in bits, across bipartition m. Notably, if the Schmidt rank of a bond tensor at bipartition m is one (i.e. the bond tensor has unit dimension), the state is separable across m. Since access to the bond tensors is required for the MPS' canonical form and also useful for these Schmidt coefficients, use of the SVD is essential, so we forgo the use of other decompositions such as the QR decomposition in our library.

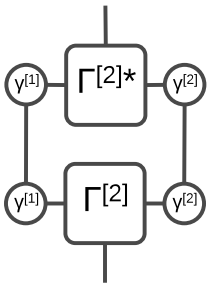
Even though we refer to this as a canonical form, an MPS in this form is not unique since



(a) Left and right canonicalisation conditions for a qudit i whose MPS tensor is labelled $\Gamma^{[i]}$. Matrices I here are the appropriately sized identities.



(b) The reduced density matrix obtained by combining Figs. 2.7a, 2.7b and 2.8b.



(c) The reduced density matrix from Fig. 2.9b if qudits $i < 2$ ($i > 2$) are in left (right) canonical form, per Fig. 2.9a.

Figure 2.9: The MPS canonicalisation conditions and their implications for the reduced density matrix. If a qudit is at the ends of the MPS, any missing bond tensors may be omitted from Figs. 2.9a and 2.9c.

the SVD itself is not unique. One such example is the trivially separable two qudit state

$$|0\rangle \otimes |0\rangle = \left(e^{i\theta}\,|0\rangle\right) \otimes \left(e^{-i\theta}\,|0\rangle\right),$$

which has different coefficients in the equivalent canonical expressions. This point is noted as an error in [36].

2.4.2 Contraction and decomposition

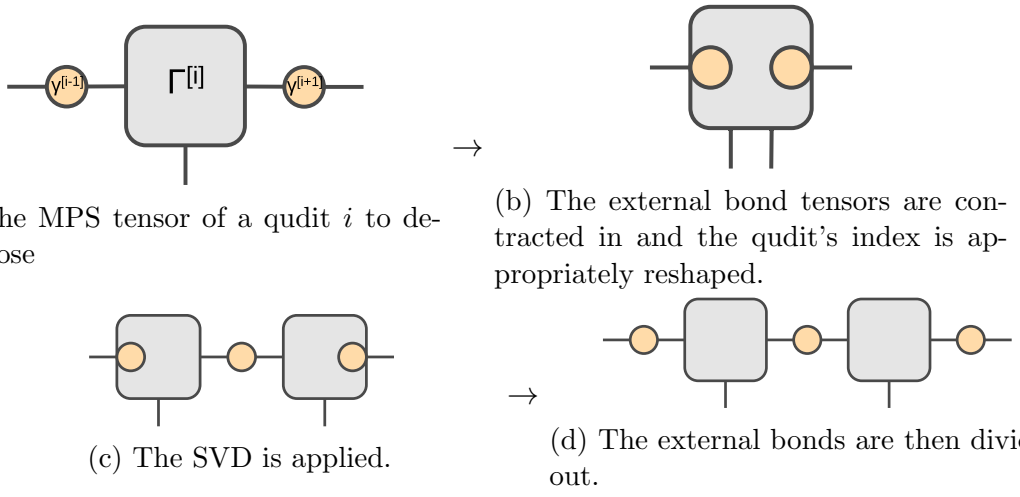
The contraction operation allows us to combine two adjacent qudits with dimensions d_1 and d_2 into a single effective qudit of dimension $d_1 d_2$. This is akin to a simple reshape of a state vector's tensor. To perform this operation, both qudit tensors are reshaped into matrices where one matrix index corresponds to the resulting free indices and the other index corresponds to the contraction indices. This initial reshape allows us to make use of optimised BLAS routines for matrix multiplication instead of slower nested loops, a choice determined on behalf of our testing and in agreement with recommendations made in [37]. The intermediate bond tensor is then contracted into the smaller of the left or right qudit's reshaped tensor to save computation time. Matrix multiplication is then performed on the two reshaped qudit tensors, and the resulting matrix is reshaped into a tensor for a qudit of dimension $d_1 d_2$. Importantly, if both qudits are in left (right) canonical form, the contracted qudit is also in left (right) canonical form. Hence, contraction maintains an MPS' canonicalisation.

Decomposition is the reverse operation of contraction, and aims to decompose a qudit of dimension $d_1 d_2$ into a left qudit of dimension d_1 and a right qudit of dimension d_2. To perform a decomposition, any existing adjacent bond tensors are first contracted into the qudit tensor. The qudit tensor is then reshaped into a matrix where one index combines the left bond index and the left subsystem's index of dimension d_1, and the other combines the right bond index

and right subsystem's index of dimension d_2. We can then perform the SVD, Eq. (2.2), to this matrix. Since the SVD is rank revealing, we can take an opportunity here to only keep tensor elements corresponding to non-zero singular values. Notably, since the bond tensors correspond to the Schmidt coefficients across the corresponding bipartition, storing only non-zero singular values allows the space requirements of a canonical MPS to scale with the amount of entanglement across each bipartition. Numerically, we only consider singular values of $m \times n$ matrix A above

$$\varepsilon \|A\|_2 \max(m, n) \tag{2.5}$$

to be non-zero, where ε is the machine epsilon of the real datatype of A (2^{-53} for double precision) and $\|A\|_2$ is the spectral norm of A, equal to A's largest singular value. This bound results from perturbations introduced in the finite precision reduction to bidiagonal form step of the SVD algorithm [31]. We then reshape the resulting matrices to restore the outer bond indices and qudit subsystem indices. Finally, since exterior bond tensors were contracted in prior to the SVD, we must perform a contraction with the element-wise reciprocal of these bond tensors to remove them. There is no issue with a division by zero at this step, since only non-zero singular values are stored. The whole decomposition process is detailed in Fig. 2.10, and contraction is naturally the reverse operation without using the external bond tensors.



(a) The MPS tensor of a qudit i to decompose

(b) The external bond tensors are contracted in and the qudit's index is appropriately reshaped.

(c) The SVD is applied.

(d) The external bonds are then divided out.

Figure 2.10: Decomposition of a qudit i's MPS tensor. If an external bond tensor is not present on account of qudit i being at the end of the MPS, it may be omitted in this process. Qudit contraction is the reverse process minus dealing with any external bonds.

Due to unitarity of the resulting matrices of an SVD, our choice to contract the adjacent bond tensors before the SVD and remove them afterwards puts the resulting left qudit in left canonical form and the resulting right qudit in right canonical form (Cf. Figs. 2.5e and 2.9a). Furthermore, if the original qudit was in left (right) canonical form, both resulting qudits must now be in left (right) canonical form, since the resulting left (right) qudit is in left (right) canonical form due to the SVD. Therefore, decomposition in this way also maintains canonicalisation of an MPS.

Contractions of more than two adjacent qudits or decompositions into more than two qudits can be achieved by successive two qudit contractions and decompositions respectively.

2.4.3 Recanonicalisation and renormalisation

A given MPS decomposition of a state vector may not be in canonical form, so we should have some operation to canonicalise such an MPS. One method is to contract together all qudits of

the MPS, effectively producing a state vector tensor, and then to sequentially decompose this state vector back into individual qudits. However, by operating on a full state vector in this intermediate step, we lose any space advantage afforded to us by an MPS representation with a rank revealing decomposition.

A more promising method of recanonicalising a general MPS that mitigates this memory cost is to perform pairwise qudit contractions and decompositions from one end of the MPS to the other end and back. We refer to a series of pairwise contractions and decompositions as a sweep, so recanonicalisation can be achieved via a sweep from the left end to the right end and back or vice-versa. Since a decomposition puts the left (right) qudit in left (right) canonical form, a right (left) sweep from the left (right) end puts every qubit except potentially the rightmost (leftmost) qudit in left (right) canonical form. However, this rightmost (leftmost) qudit is guaranteed to be in left (right) canonical form if the entire state is normalised correctly to $\langle\psi|\psi\rangle = 1$. Since we only work on pairs of qudits at a time, we do not incur as great a memory cost as contracting into an entire state vector.

If a state is not normalised correctly or becomes unnormalised (possibly due to a build up of numeric imprecision), we can still perform a complete left and right sweep to produce an MPS in an unnormalised canonical form. In this form, a bond tensor $\lambda^{[m]}$ at bipartition m still contains the Schmidt coefficients of m, but their squares no longer sum to one:

$$\begin{aligned}\sum_{\alpha_m} \left|\gamma^{[m]}_{\alpha_m}\right|^2 &= \langle\psi|\psi\rangle \\ &\neq 1.\end{aligned}$$

Therefore, we should first rescale the bond tensor for each bipartition m via

$$\gamma^{[m]}_{\alpha_m} \leftarrow \frac{\gamma^{[m]}_{\alpha_m}}{\sqrt{\langle\psi|\psi\rangle}}.$$

To produce the correct overall normalisation, qudit tensor $\Gamma^{[q]}$ for each interior qudit q (i.e. q is not at an end of the MPS chain) is updated via

$$\Gamma^{[q]i_q}_{\alpha_q\alpha_{q+1}} \leftarrow \Gamma^{[q]i_q}_{\alpha_q\alpha_{q+1}} \sqrt{\langle\psi|\psi\rangle},$$

where i_q is the qudit index, α_q is the left bond index and α_{q+1} is the right bond index. Since there is exactly one more bond than there is interior qudit, the overall affect of these rescalings is to take

$$|\psi\rangle \leftarrow \frac{|\psi\rangle}{\sqrt{\langle\psi|\psi\rangle}},$$

thus normalising and canonicalising the MPS correctly. Furthermore, these rescalings do not require any MPI communication (apart from initially determining $\langle\psi|\psi\rangle$ from a bond in an unnormalised canonical MPS) and are easily vectorised, so this extra normalisation step has minimal time overhead.

2.4.4 Truncation

We have established that the use of the rank revealing SVD allows a space advantage for a canonical MPS representation of a quantum state with low Schmidt rank (i.e. low entanglement) across each bipartition when compared to the full state vector. If we consider some bond m in a canonical MPS and contract all qudits left of m into subsystem A and all qudits right of m

into subsystem B, we arrive at the Schmidt decomposition

$$|\psi\rangle = \sum_{\alpha_m=1}^{\chi} \gamma_{\alpha_m}^{[m]} \left|\Gamma_{\alpha_m}^{[A]}\right\rangle \otimes \left|\Gamma_{\alpha_m}^{[B]}\right\rangle, \tag{2.6}$$

where χ is the numerical Schmidt rank across m, remembering that we only chose to consider tensor elements corresponding to non-zero singular values according to the threshold defined in Eq. (2.5). Numerical implementations of the SVD produce the (non-negative, real) singular values in descending order, so we let $\gamma_{\alpha_m}^{[m]}$ decrease with α_m. Therefore, the Schmidt decomposition in Eq. (2.6) with decreasing singular values $\gamma_{\alpha_m}^{[m]}$, gives rise to a natural lossy compression scheme to approximate the state $|\psi\rangle$ by extending the sum to some compressed Schmidt rank $\xi < \chi$. In the context of an MPS, this process of reducing a bond's dimension is referred to as a truncation. It produces an MPS with smaller memory requirements at the cost of fidelity to the original state. We also note that truncation in this manner requires the use of the SVD to provide access to the Schmidt coefficients through the singular values.

Directly following the truncation of a bond between left qudit q_1 and right qudit q_2, q_1 is no longer in right canonical form and q_2 is no longer in left canonical form. The whole MPS can be returned to a canonical form by performing a left sweep from q_1 and a right sweep from q_2. However, this canonical form is unnormalised, since truncation of the bond between q_1 and q_2 removes some Schmidt coefficients. We can apply a renormalisation at this point to produce a canonical MPS with correct normalisation.

It is important to note that truncation to a canonical MPS is not a purely local operation, i.e. one involving the two qudits adjacent to the truncated bond. Truncation of a bond of Schmidt rank χ to a truncated rank ξ can be thought of as receiving

$$|\psi_{\text{trunc}}\rangle = \sum_{\alpha_m=1}^{\xi<\chi} \gamma_{\alpha_m}^{[m]} \left|\Gamma_{\alpha_m}^{[A]}\right\rangle \otimes \left|\Gamma_{\alpha_m}^{[B]}\right\rangle$$

as the result of some measurement (before renormalisation). Since a measurement should in general serve to collapse entanglement throughout a state, sweeps outward from the truncated bond are required to propagate this collapse. Numerical tests show that renormalising the truncated bond alone, as erroneously suggested in [36], is not enough to renormalise, let alone recanonicalise an MPS.

Canonicalised truncation in this method is a per-bond operation. If entanglement is introduced into multiple bonds simultaneously (perhaps through the application of a gate operation on multiple qudits as per subsection 2.4.6) through an increase in the Schmidt ranks, the user must choose some ordering of these bonds to perform canonicalised truncations on. We also provide the option of performing an uncanonicalised truncation, where bonds can simply be truncated and the recanonicalisation sweeps and renormalisation operation can be applied just prior to some operation requiring a canonical MPS.

2.4.5 Measurement and reset

Measurement is the method by which we extract any results from a quantum algorithm running on a quantum computer. In section 2.2, we required the reduced density matrix of a qudit subsystem in order to measure it. Furthermore, in subsection 2.4.1, we showed how canonicalisation allows us to calculate a qudit's reduced density matrix locally. Combining these points, the first step of performing a measurement of qudit q is to ensure that every qudit left (right) of q is in left (right) canonical form. This is automatically the case if the MPS is in canonical form, but can be achieved by sweeping inward from the ends at least up to q. We

can then calculate the diagonal elements p_m of the reduced density matrix of q locally. Since the p_m correspond to probabilities of measuring value m from qudit q, we can sample from this discrete distribution to simulate a measurement result of some value m.

To put the MPS into a state where m was measured from qudit q, we temporarily shrink q to a one dimensional qudit subsystem corresponding to the measured value m with correct normalisation. That is, if m is the result of our measurement, we project

$$\Gamma^{[q]}_{\alpha_q\alpha_{q+1}} \leftarrow \frac{\Gamma^{[q]m}_{\alpha_q\alpha_{q+1}}}{\sqrt{p_m}}. \tag{2.7}$$

Since qudits left (right) of q were already in left (right) canonical form prior to this projection, we only need to perform sweeps outward from q in both directions to canonicalise the MPS (if we wish to canonicalise it). Since measurement collapses entanglement throughout the state, these outward sweeps serve to propagate this collapse of entanglement, in a similar fashion to the outward sweeps of truncation.

We chose to shrink q to a qudit of dimension one prior to the sweeps to avoid contraction and decomposition operations using a tensor that would otherwise have zeros for each element not corresponding to m. This method results in exceptional time and space savings, especially if the dimensionality of q was very large (i.e. if q consists of many qubits). However, since a one level qudit subsystem is not computationally useful, we must expand q's tensor back to full dimensionality if q is to be used in further operations. That is, we update q's tensor so that

$$\Gamma^{[q]i_q}_{\alpha_q\alpha_{q+1}} \leftarrow \begin{cases} \Gamma^{[q]}_{\alpha_q\alpha_{q+1}}, & i_q = m \\ 0, & i_q \neq m \end{cases}. \tag{2.8}$$

Otherwise, if q is not needed for further operations, we can contract the unexpanded, one dimensional qudit q into an adjacent qudit, effectively removing it since the measurement makes q separable from the rest of the system. We choose to contract into the adjacent qudit with the smaller tensor to minimise computation time.

If we wish to reset a qudit q to a certain value s, we perform an unexpanded qudit measurement of q with or without the subsequent canonicalisation, producing measurement result m. We then perform some classical switching on q that maps $|m\rangle \to |s\rangle$. This can be accomplished via an alternate expansion to Eq. (2.8) where the unexpanded tensor for m is placed into s:

$$\Gamma^{[q]i_q}_{\alpha_q\alpha_{q+1}} \leftarrow \begin{cases} \Gamma^{[q]}_{\alpha_q\alpha_{q+1}}, & i_q = s \\ 0, & i_q \neq s \end{cases},$$

following from Eq. (2.7).

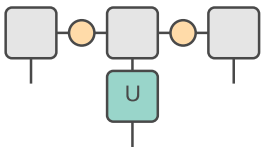
2.4.6 Gate operations

In order to simulate quantum circuits, we must be able to apply quantum gates to our MPS. As mentioned in section 2.2, quantum gates are unitary operations that contract onto a state vector when applied. For single qudit gates, the generalisation to an MPS is simple: the gate's unitary matrix in the computational basis is simply contracted to the specified qudit's tensor, whilst maintaining any left or right canonicalisation of the qudit and hence any canonicalisation of the entire MPS. The schematic for this process is given by Fig. 2.11a.

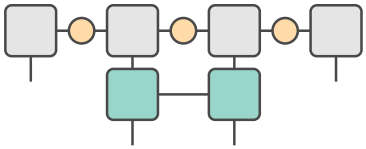
Gates on multiple qudits are easily applied to a state vector through tensor contraction and appropriate index bookkeeping. Since an MPS lays the qudit tensors out in a linear chain, it is most straightforward to apply a multiple qudit gate to consecutive qudits on an MPS. To apply

a gate on n consecutive qudits, there are two options. The first is to contract the n qudits into a single qudit, apply the gate as a single qudit unitary operation and then decompose back into n qudits. This method maintains canonicalisation since the contraction, single qudit gate operation and decomposition all do.

The second method involves decomposing the gate itself into a matrix product operator (MPO), which can then be contracted directly onto the n qudits in the MPS (Fig. 2.11b). This contraction alone does not maintain canonicalisation, but if all n qudits were initially in left (right) canonical form, then a sweep right (left) over just these n qudits will return them all to left (right) canonical form. A single such sweep can be performed as the MPO is contracted onto each subsequent qudit of the MPS, in order to leverage the rank revealing SVD and minimise space usage during the operation. When the total dimensionality d of the n qudits is very large, the latter method may have lower space requirements since only pairs of qudits are contracted at once. However, in this case, finding an MPO representation of the $d \times d$ unitary matrix through tensor decomposition may prove computationally challenging. We propose an efficient method to apply common linear nearest neighbour (LNN) gates in section 2.5. Otherwise if d is reasonable, as is the case for two qubit ($d = 4$) gates such as the Swap gate, the former method is faster due to needing to perform fewer tensor contractions.



(a) Application of gate U to the second site of an MPS. If U acts on multiple, adjacent subsystems, they should first be contracted into a single qudit for this method.



(b) Otherwise, U itself might be decomposed into an MPO and applied to each relevant site.

Figure 2.11: Methods for applying gate operations to MPSs

When applying non-LNN gates (i.e. multiple qudit gates on non-consecutive qudits) to an MPS, there are also two strategies. If we wish to apply an n qudit gate U to n qudits in our MPS that lie across a range including an additional k qudits, we can regard an expanded U as an $n + k$ qudit gate that applies U to the n core qudits and the identity operator I to the k extra qudits. Given an MPO decomposition of U, tensors for the separable identity operators can be inserted into this MPO in positions corresponding to the k extra qudits in the MPS. This expanded MPO for U can then be applied as an LNN gate as mentioned above.

The second strategy for applying a non-LNN gate U is to use Swap operations on two adjacent qudits

$$\textsc{Swap}: \quad |a\rangle \otimes |b\rangle \to |b\rangle \otimes |a\rangle\,, \tag{2.9}$$

where $|a\rangle$ is a basis ket of one qudit's Hilbert space and $|b\rangle$ is a basis ket of the other's. Through enough of these Swap gates, which can be applied in either method as two-qudit LNN gates, qudits in the linear chain of the MPS can be rearranged so that U can be applied as an LNN gate. Optimising Swap placement is crucial for performance; at least one such algorithm [38] aims to map general quantum circuits to LNN architectures with as few Swap gates as possible. Optimising specifically for MPS simulation performance may be even more difficult due to the non-linear time and space complexities of qudit contraction and decomposition and not being able to easily query a bond's resulting dimension prior to operating a Swap.

2.5 General controlled gate operations for MPS

One of the most common multiple qudit gate structures is the controlled gate operation. Consider first a single qubit gate such as the X gate

$$|0\rangle \to |1\rangle\,,$$
$$|1\rangle \to |0\rangle\,,$$

the quantum equivalent of the classical NOT gate in the computational basis. A controlled X gate, otherwise known as the CNOT gate, applies the X gate to a qubit based on the state of another:

$$|0\rangle \otimes |0\rangle \to |0\rangle \otimes |0\rangle\,,$$
$$|0\rangle \otimes |1\rangle \to |0\rangle \otimes |1\rangle\,,$$
$$|1\rangle \otimes |0\rangle \to |1\rangle \otimes |1\rangle\,,$$
$$|1\rangle \otimes |1\rangle \to |1\rangle \otimes |0\rangle\,.$$

Here, the state of the first qubit is used to conditionally apply the X gate on the second qubit: if the first qubit is in the $|1\rangle$ state, then the X gate is applied to the second qubit; otherwise the identity operation is applied to the second qubit. The state of the control qubit is left unchanged. This is a form of binary control, where each state for a control qudit is individually either 'on' or 'off', specifying whether or not to apply the attached gate. For example, a five-level control qudit might apply the main gate operation in states $|0\rangle$, $|2\rangle$ or $|4\rangle$, whilst applying the identity in states $|1\rangle$ or $|3\rangle$. When there are multiple binary control qudits, the main gate operation is only applied when all control qudits are in an 'on' state. If there are multiple simultaneously controlled gate operations, they are treated as a single operation: that is, if the control qudits take an 'on' state, then all attached gate operations are applied to their respective qudits. Otherwise, if any control qudit takes an 'off' state, none of the attached gates are applied.

For an MPS, we are interested in being able to apply LNN restricted versions of these general binarily controlled gate operations. Specifically, we require that not only the entire controlled gate operation be LNN, but also that the main gates (i.e. the gates binarily controlled by the control qudits) be LNN. Fig. 2.12 contains an example of such a gate. If the main gates are each LNN but the control qudits cause the entire gate structure not to be LNN, identity gate operations I can be inserted at each qudit gap. In this way, we may simulate, for example, a CNOT gate on distant qubits as a controlled $I \otimes \ldots \otimes I \otimes X$ gate. However, this does not provide a simple method of applying a long range SWAP (Eq. (2.9)) to two qudits.

To review the options detailed in subsection 2.4.6 of applying a gate to an MPS, we could either contract the required qudits into a single qudit that we then apply the gate to, or decompose a matrix representation of the gate into an MPO and contract that onto the MPS. Both methods have space limitations when the total dimensionality of the required qudits is very large: the former in contracting the qudits together and the latter in decomposing a large matrix representing the gate operation. However, it is possible to apply an LNN generalised binarily controlled gate (LNNGBCG) whilst avoiding complete qudit contraction or operator decomposition.

We propose the following rules for obtaining a resulting MPS after such an LNNGBCG is applied. From these, an MPO representation of the gate can easily be obtained. We require that any main gates are applied as single qudit operations: for example, a two qubit SWAP requires the two adjacent qubits be contracted. Moreover, in the interest of space and time usage, the following traits are desirable:

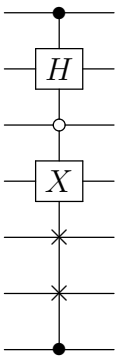


Figure 2.12: An example LNNGBCG on seven qubits. The core gates, H, X and Swap, are each LNN, and the entire gate is also LNN. Black (white) dots represent controls on state $|1\rangle$ ($|0\rangle$). Our aim is to apply gates like these to MPSs without writing out their full matrix representations.

- The derived qudit tensors should be block sparse whenever possible, since zeroing a contiguous region of a distributed tensor is fast and requires no MPI communication.
- The non-zero regions of a derived qudit tensor should only include slices from the original qudit's tensor or, for qudits on which the main gates operate, the result of the main gates on these slices. This ensures that only useful computation is done to find the resulting MPS representation, limiting us through memory and communication speed.
- If an MPO representation of the entire LNN gate is provided, the MPS resulting from our rules should not have a greater space usage than an MPS resulting from contraction with the given MPO.

With regard to the third point, we observed that the Schmidt operator rank across some bipartition of an LNNGBCG MPO, obtained through a rank revealing decomposition such as the SVD, is at most 3 if there exists at least one main gate and one control on both sides of the bipartition; otherwise, it is at most 2. For example, the LNNGBCG of Fig. 2.12 has Schmidt operator rank 2 between the H and the above control, but rank 3 between it and the below control. This simple classification gives rise to a general operator Schmidt rank structure of LNNGBCGs. On the left and the right sides of the LNNGBCG, there must be maximal and contiguous regions of sites that consist entirely of either main gates or controls. Within the left (right) region, the operator Schmidt rank is 2, since all sites to the left (right) of any bipartition in this region are either main gates or controls. Optionally, there may also be a contiguous middle region for the remaining sites. If this region has more than one site, any bipartitions within it have an operator Schmidt rank of 3. If the left and right partitions consist of sites of the same classification (i.e. main gate or control), this middle region is required. Otherwise, every site would be a control, which would result in an invalid gate, or every site would be a main gate, which would not be an LNNGBCG; rather, it would be applied via individual single qudit gates. Therefore, to obtain a complete description of the results of applying an LNNGBCG to an MPS, we must provide a qudit's resulting tensor for

1. the leftmost LNNGBCG site, where the MPO's left and right Schmidt ranks are 1×2 at this site,
2. other sites in the left region, where the left and right operator Schmidt ranks are 2×2,
3. any middle region sites, where
   (a) a single middle region site has operator Schmidt ranks of 2×2

(b) or if there are multiple middle region sites,

i. the leftmost site of the middle region has ranks 2×3,

ii. any interior sites of the middle region has ranks 3×3 and

iii. the middle region's rightmost site has ranks 3×2,

4. other sites of the right region, with ranks 2×2, and finally

5. the rightmost site, where the rank is 2×1,

for both cases of an LNNGBCG site's classification.

Conventionally, we write the elements of a single n-dimensional qudit's state $|\psi\rangle$ as a column vector in the computational basis which may be transformed by some operator U expressed as a square matrix in the same basis:

$$U \begin{bmatrix} \psi_1 \\ \psi_2 \\ \dots \\ \psi_n \end{bmatrix} = \begin{bmatrix} U_{1\alpha}\psi_\alpha \\ U_{2\alpha}\psi_\alpha \\ \dots \\ U_{n\alpha}\psi_\alpha \end{bmatrix}. \tag{2.10}$$

For the tensor $\Gamma^{[q]}$ of some qudit q in an MPS whose elements are indexed as $\Gamma^{[q]i_q}_{\beta_q\beta_{q+1}}$ where β_q and β_{q+1} are the left and right bond indices respectively, the contraction of an operator U onto q generalises from Eq. (2.10) to a 'vector of matrices':

$$\begin{bmatrix} \Lambda^{[q]1} \\ \Lambda^{[q]2} \\ \dots \\ \Lambda^{[q]n} \end{bmatrix} \equiv U \begin{bmatrix} \Gamma^{[q]1} \\ \Gamma^{[q]2} \\ \dots \\ \Gamma^{[q]n} \end{bmatrix} = \begin{bmatrix} U_{1\alpha}\Gamma^{[q]\alpha} \\ U_{2\alpha}\Gamma^{[q]\alpha} \\ \dots \\ U_{n\alpha}\Gamma^{[q]\alpha} \end{bmatrix},$$

where $\Lambda^{[q]}$ is the result of applying U to q's tensor.

Using this notation, if the resulting tensor for q after the full LNNGBCG is applied is $\Xi^{[q]}$, then the main gate site rules are

$$\begin{aligned}
1: \quad \Xi^{[q]i_q} &= \begin{bmatrix} \Gamma^{[q]i_q} & \Lambda^{[q]i_q} \end{bmatrix}, \\
2,\ 3\text{a},\ 4: \quad \Xi^{[q]i_q} &= \begin{bmatrix} \Gamma^{[q]i_q} & 0 \\ 0 & \Lambda^{[q]i_q} \end{bmatrix}, \\
3(\text{b})\text{i}: \quad \Xi^{[q]i_q} &= \begin{bmatrix} \Gamma^{[q]i_q} & \Gamma^{[q]i_q} & 0 \\ \Gamma^{[q]i_q} & 0 & \Lambda^{[q]i_q} \end{bmatrix}, \\
3(\text{b})\text{ii}: \quad \Xi^{[q]i_q} &= \begin{bmatrix} \Gamma^{[q]i_q} & 0 & 0 \\ 0 & \Gamma^{[q]i_q} & 0 \\ 0 & 0 & \Lambda^{[q]i_q} \end{bmatrix}, \\
3(\text{b})\text{iii}: \quad \Xi^{[q]i_q} &= \begin{bmatrix} \Gamma^{[q]i_q} & 0 \\ 0 & \Gamma^{[q]i_q} \\ 0 & \Lambda^{[q]i_q} \end{bmatrix}, \\
5: \quad \Xi^{[q]i_q} &= \begin{bmatrix} \Gamma^{[q]i_q} \\ \Lambda^{[q]i_q} \end{bmatrix},
\end{aligned}$$

where the respective matrices are concatenated, and 0 is the zero matrix of the appropriate size here. For the control sites, if u is an 'off' value and c is an 'on' value of the respective control

in the LNNGBCG, we have the rules

$$
\begin{aligned}
1: &\quad \Xi^{[q]u} = \begin{bmatrix}\Gamma^{[q]u} & 0\end{bmatrix}, & 1: &\quad \Xi^{[q]c} = \begin{bmatrix}0 & \Gamma^{[q]c}\end{bmatrix},\\
2: &\quad \Xi^{[q]u} = \begin{bmatrix}\Gamma^{[q]u} & 0\\ \Gamma^{[q]u} & 0\end{bmatrix}, & 2: &\quad \Xi^{[q]c} = \begin{bmatrix}\Gamma^{[q]c} & 0\\ 0 & \Gamma^{[q]c}\end{bmatrix},\\
3\text{a}: &\quad \Xi^{[q]u} = \begin{bmatrix}\Gamma^{[q]u} & 0\\ 0 & 0\end{bmatrix}, & 3\text{a}: &\quad \Xi^{[q]c} = \begin{bmatrix}0 & 0\\ 0 & \Gamma^{[q]c}\end{bmatrix},\\
3(\text{b})\text{i}: &\quad \Xi^{[q]u} = \begin{bmatrix}\Gamma^{[q]u} & \Gamma^{[q]u} & 0\\ 0 & 0 & 0\end{bmatrix}, & 3(\text{b})\text{i}: &\quad \Xi^{[q]c} = \begin{bmatrix}\Gamma^{[q]c} & 0 & 0\\ 0 & 0 & \Gamma^{[q]c}\end{bmatrix},\\
3(\text{b})\text{ii}: &\quad \Xi^{[q]u} = \begin{bmatrix}\Gamma^{[q]u} & \Gamma^{[q]u} & 0\\ 0 & 0 & 0\\ 0 & 0 & 0\end{bmatrix}, & 3(\text{b})\text{ii}: &\quad \Xi^{[q]c} = \begin{bmatrix}\Gamma^{[q]c} & 0 & 0\\ 0 & \Gamma^{[q]c} & 0\\ 0 & 0 & \Gamma^{[q]c}\end{bmatrix},\\
3(\text{b})\text{iii}: &\quad \Xi^{[q]u} = \begin{bmatrix}\Gamma^{[q]u} & 0\\ 0 & 0\\ 0 & 0\end{bmatrix}, & 3(\text{b})\text{iii}: &\quad \Xi^{[q]c} = \begin{bmatrix}0 & 0\\ \Gamma^{[q]c} & 0\\ 0 & \Gamma^{[q]c}\end{bmatrix},\\
4: &\quad \Xi^{[q]u} = \begin{bmatrix}\Gamma^{[q]u} & \Gamma^{[q]u}\\ 0 & 0\end{bmatrix}, & 4: &\quad \Xi^{[q]c} = \begin{bmatrix}\Gamma^{[q]c} & 0\\ 0 & \Gamma^{[q]c}\end{bmatrix},\\
5: &\quad \Xi^{[q]u} = \begin{bmatrix}\Gamma^{[q]u}\\ 0\end{bmatrix}, & 5: &\quad \Xi^{[q]c} = \begin{bmatrix}0\\ \Gamma^{[q]c}\end{bmatrix}.
\end{aligned}
$$

For bipartitions of operator Schmidt rank bounded by 2 (3), the corresponding bond vectors of the MPS are tiled twice (thrice).

These rules satisfy the first and second desirable traits from above: they are block sparse and only rely on values from the original qudit tensor or from the qudit tensor after applying a main gate. Furthermore, they are consistent with our assertion that the operator Schmidt rank across any bipartition of an LNNGBCG is bounded above by 2 or 3 depending on whether or not the bipartition has at least one main gate and one control on both sides. Though we haven't proven opimality of this classification, we motivate it by claiming that a rank of 3 results from controls on each side of a bipartition controlling main gates on the opposite side, whilst a rank of 2 results from controls on only one side of a bipartition controlling gates on the opposite side.

Application of an LNNGBCG does not maintain canonicalisation of the MPS. The MPS can be recanonicalised as mentioned in subsection 2.4.6 for general MPOs, where the contraction and decomposition operations of a single sweep are performed as each individual qudit and bond tensor is subsequently transformed.

2.6 Density matrices in MPS

Just as the tensor network principles allow us to decompose a state vector into an MPS, a general density matrix

$$\rho = \sum_i p_i \, |\psi_i\rangle\langle\psi_i| \tag{2.11}$$

is Hermitian with unit trace and may be decomposed into a similar LNN tensor network as shown in Fig. 2.13. Since ρ is also an operator which can be applied to some arbitrary state

$|\psi\rangle$ as

$$\rho\,|\psi\rangle = \sum_i \langle\psi_i|\psi\rangle\,|\psi_i\rangle\,,$$

the tensor network decomposition of a density matrix is, infact, an MPO. However, we will specifically refer to these as matrix product density operators (MPDOs) for density matrices. Eq. (2.11) implies a tensor product of state vectors, so if an MPDO is to follow the LNN form of an MPS, each qudit q of the MPDO holds some tensor $\Theta^{[q]}$ with four indices as $\Theta^{[q]i_q j_q}_{\alpha_q \alpha_{q+1}}$, where α_q and α_{q+1} are the familiar left and right bond indices akin to those in an MPS and i_q and j_q correspond to q's indices in the overall density matrix. The bond tensors are similarly vectors resulting from the SVD as our decomposition.

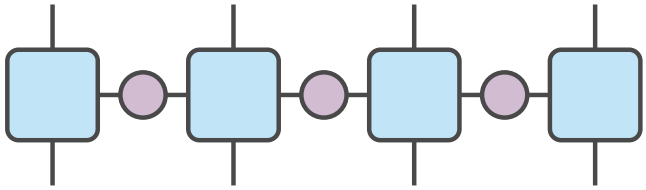


Figure 2.13: Tensor network for an MPDO of four qudits. Cf. Fig. 2.7a.

In subsection 2.4.1, we discussed a canonical form for an MPS which allowed us to obtain a qudit's reduced density matrix locally and hence perform single qudit measurements locally. This canonical form resulted from unitarity of the SVD and normalisation of a state, $\langle\psi|\psi\rangle = 1$, which can also be seen if we consider the density matrix ρ of the pure state $|\psi\rangle$ also contracted with its dual:

$$\begin{aligned}\mathrm{tr}(\rho\rho^\dagger) &= \langle\psi|\psi\rangle\,\langle\psi|\psi\rangle \\ &= 1.\end{aligned}$$

Though we can still choose to use the SVD, no similar normalisation for a general, potentially mixed density matrix exists, which is unsurprising given that this quantity is also referred to as a state's purity:

$$\begin{aligned}\mathrm{tr}(\rho\rho^\dagger) &= \mathrm{tr}(\rho^2) \\ &\leq 1\end{aligned}$$

where the equality results from the density matrix being Hermitian. Therefore, a similar canonical form for our MPDOs cannot be used to locally obtain a qudit's reduced density matrix or perform a single qudit measurement. We are required to trace over all other indices (in a similar fashion to Fig. 2.7b) in order to obtain a qudit's reduced density matrix and measure it. However, we may still define a canonical form for MPDOs similar to that of MPSs, the only difference being that a 'loop' (like in Fig. 2.9a) has an extra contracted index line corresponding to the density matrix's extra set of indices, and that the rightmost (leftmost) qudit will have a left (right) loop equal to $\mathrm{tr}(\rho^2)$ instead of 1 for all other qudits. The operations to maintain this canonical form, including sweeps, have the benefit of minimising the bond tensors' dimensions to their respective operator Schmidt ranks at each bipartition.

A gate operation U on an MPDO now requires a contraction with U and its adjoint:

$$\rho \leftarrow \sum_i p_i U\,|\psi\rangle\langle\psi|\,U^\dagger = U\rho U^\dagger,$$

as demonstrated in Fig. 2.14. Each of the strategies for applying a gate to an MPS are applicable here: qudits can be contracted and then gate U applied, or U can be decomposed into an MPO and applied to multiple qudits. The LNNGBCGs of section 2.5 can also be adapted for use MPDOs with further bookkeeping.

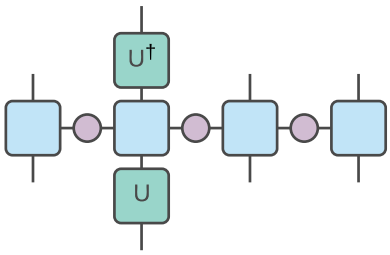


Figure 2.14: Operator U is applied to the second qudit in an MPDO.

Measurement of a single qudit q may be performed by obtaining the probabilities p_m of measuring a value m from the diagonal elements of q's reduced density matrix. Since an MPDO's canonical form does not make this a local operation, we must trace over all other qudits to obtain these probabilities, from which we sample to choose a measurement result m. If the qudit's tensor is $\Theta^{[q]}$, it is updated by

$$\Theta^{[q]i_q j_q}_{\alpha_q \alpha_{q+1}} \leftarrow \begin{cases} \Theta^{[q]mm}_{\alpha_q \alpha_{q+1}}/p_m, & i_q = j_q = m \\ 0, & \text{otherwise} \end{cases},$$

where optimisations similar to those for an MPS in subsection 2.4.5 can be made by reducing q to a single dimensional qudit if it is no longer required in the circuit. Sweeps outward from q are then required to return the MPDO to canonical form and collapse entanglement.

2.6.1 Error model simulations with MPDOs

If we only consider MPDO analogues for the operations we have defined for MPSs, then a pure state MPDO will remain pure. In this case, it is more space efficient to use an MPS to simulate pure states. An additional operation, exclusive to MPDOs, is the application of a quantum noise channel to a qudit. These are single qudit operations that introduce some amount noise to a pure state, resulting in a mixed state. Simulating these operations allows one to examine the effects that the noise present in a physical quantum machine might have on the results from a quantum circuit. One example of a noise channel is the bit-flip of probability p on a single qubit:

$$\rho \leftarrow (1-p)\rho + pX\rho X^\dagger, \tag{2.12}$$

which may be interpreted as applying the bit-flip X gate as a noise operation with probability p. A single qubit's density matrix ρ is thus updated via

$$\rho \equiv \begin{bmatrix} \rho_{00} & \rho_{01} \\ \rho_{10} & \rho_{11} \end{bmatrix} \leftarrow (1-p)\begin{bmatrix} \rho_{00} & \rho_{01} \\ \rho_{10} & \rho_{11} \end{bmatrix} + p\begin{bmatrix} \rho_{11} & \rho_{10} \\ \rho_{01} & \rho_{00} \end{bmatrix}.$$

In an MPDO where qubit q has tensor $\Theta^{[q]}$, the update applies blockwise:

$$\Theta^{[q]} = \begin{bmatrix} \Theta^{[q]00} & \Theta^{[q]01} \\ \Theta^{[q]10} & \Theta^{[q]11} \end{bmatrix} \leftarrow (1-p)\begin{bmatrix} \Theta^{[q]00} & \Theta^{[q]01} \\ \Theta^{[q]10} & \Theta^{[q]11} \end{bmatrix} + p\begin{bmatrix} \Theta^{[q]11} & \Theta^{[q]10} \\ \Theta^{[q]01} & \Theta^{[q]00} \end{bmatrix}.$$

Applying a noise channel is a discrete operation used to model the effects of introducing a specific kind of quantum noise to a system. When modelling noise, the choice and placement of a noise channel in a circuit must be considered. Bit-flip channels, for example, may be applied to each qubit in a quantum circuit immediately following initialisation or preceding measurement to model initialisation and measurement errors respectively.

One more example of a noise channel is the depolarising channel, which updates the density matrix ρ of a general n-dimensional qudit as

$$\rho \leftarrow (1-p)\rho + p\,\mathrm{tr}(\rho)\frac{I_n}{n} \tag{2.13}$$

where I_n is the $n \times n$ identity matrix. Here, $I_n/n = \sum_{i=0}^{n-1} \frac{1}{n} |i\rangle\langle i|$ is the density matrix of the fully mixed or fully depolarised state and $\mathrm{tr}(\rho) = 1$ is a convenient way to relate its elements to ρ's. Therefore, the depolarising channel can be interpreted as depolarising a state with probability p, and a qubit q with tensor $\Theta^{[q]}$ in an MPDO is updated via

$$\Theta^{[q]} \leftarrow (1-p)\Theta^{[q]} + p\frac{1}{2}\begin{bmatrix} \Theta^{[q]00} + \Theta^{[q]11} & 0 \\ 0 & \Theta^{[q]00} + \Theta^{[q]11} \end{bmatrix},$$

which generalises to higher dimensions n by summing over more diagonal blocks $\Theta^{[q]i_q i_q}$. These depolarising channels are often placed after the usual gate operations.

Since noise channels can be expected to decohere quantum states, entanglement in our MPDO should accordingly collapse. Therefore, sweeps outward in both directions from the qudit to which a noise channel is applied are required to propagate this collapse and recanonicalise an MPDO, in a similar fashion to single qudit measurement.

Several more standard noise channels have also been defined, and can be found in [39]. QCMPS includes functionality for also applying user-pecified noise channels.

2.6.2 Alternate approaches for density matrix simulations

A straightforward approach to reproducing the measurement statistics of a density matrix simulated with noise channels is to sample from sufficiently many state vectors. For example, if a bit-flip channel of Eq. (2.12) is applied to qubit q, we could emulate it with a state vector by applying the X gate to q with probability p and averaging over the measurement probabilities for a sufficiently large sample of such state vector runs. Since the space usage of an MPS is significantly lower than its MPDO counterpart, we are effectively trading this space saving for the additional time required to sufficiently sample noise outcomes. If a significant amount of noise is introduced, enough entanglement in an MPDO may be collapsed through canonicalisation, and an MPS may require far too many samples to sufficiently characterise the error model. In this case, it makes sense to use the MPDO formalism. Furthermore, to average over the measurement probabilities for a subsystem q of an MPS, the diagonal elements of q's reduced density matrix are required. The constituent qudit subsystems of q may need to be swapped and contracted to form a qudit for q to obtain this reduced density matrix. If the dimensionality of q is very large, there may be space limitations in even just storing the measurement probabilities; if q is the entire system, then a full contraction is required. However, with an MPDO, each qudit subsystem can be measured sequentially without full contraction, allowing us to sample from the correct measurement probability distribution without incurring this memory cost.

2.7 Comparison to existing quantum simulators

One of the simplest and most complete quantum simulator packages is the open source Quantum Toolbox in Python (QuTiP) [18]. Built upon NumPy [26], it also interfaces with the rest of the SciPy stack to provide rich plotting capabilities through Matplotlib [40]. QCMPS in its current form does not support any visualisation capabilities, but may in the future. Though QuTiP relies on the simpler state vector and density matrix formalisms in its calculations as opposed to our MPS or MPDO methods and is limited to shared memory parallelism, it does internally store its tensors sparsely, leading to decreased time and space usage in certain situations. The tensors in QCMPS are stored densely, since Elemental [32] only supports dense, distributed SVD.

QCMPS was built upon Elemental in order to utilise distributed memory architectures to overcome the space limitations of shared memory systems. One other distributed memory quantum simulator is *q*H*i*PSTER [19], a similar simulator described by Boixo et al. in [17]. This is solely a state vector simulator, without any density matrix, MPS or MPDO functionality. Furthermore, it is limited to single-qubit gates and singly-controlled single-qubit gates such as the CNot, forcing the user to decompose their circuits. In some sense, this extra step is analogous to the extra Swap gates required to map a general gate to an MPS or MPDO. While an MPS may have a space and time advantage over a state vector when a state has low entanglement, its space complexity is equal ($\mathcal{O}(2^n)$ for an n qubit system) to a state vector's for sufficiently entangled states. In this case, a state vector simulator is advantageous timewise, since it has no need to apply expensive matrix decompositions such as the SVD that may or may not scale well with additional CPU cores. Indeed, simple controlled or uncontrolled single qubit gates as tensor contractions to a state vector are extremely parallelisable, and *q*H*i*pster makes use of vectorisation, multithreading and cache locality optimisations to increase its performance.

Most libraries for manipulating MPSs focus not on quantum circuit simulations, but rather different algorithms such as the density matrix renormalisation group (DMRG) [22] or time-evolving block decimation (TEBD) [23] for general quantum many-body problems or problems in quantum chemistry. Though outside the scope of this thesis, their feature requirements are slightly different to our circuit simulator: we place more emphasis on strict canonicalisation to allow local measurement of any qudit at any time, requiring use of the SVD. Two noteworthy tensor network libraries are ITensor [41] and Tensor Network Theory (TNT) [42]. ITensor is a purely shared memory library, while TNT is single-node GPU-accelerated. These are general-purpose tensor network libraries that can be used to implement the operations of QCMPS on single node architectures.

In summary, QCMPS uniquely combines distributed memory parallelism for one-dimensional quantum circuit simulation with original features such as LNNGBCGs and noisy MPDO functionality.

Chapter 3

Shor's Algorithm Simulations

In order to benchmark our MPS library QCMPS, we apply a similar technique to [24] with additional optimisations to perform a simulation of Shor's algorithm [2] for factoring semiprime integers. We will present the optimisations used to factor a semiprime of 20 binary digits through a simulation of a high-level quantum circuit for Shor's algorithm requiring 60 qubits. For comparison, a state vector for 60 qubits requires us to store 2^{60} complex scalars. At double precision (64 bit real and 64 bit imaginary components), this requires a minimum storage capacity of 16 EiB, more than the available memory of any currently existing supercomputer today [15]. The MPS approach from [24] of simulating Shor's algorithm is able to reduce the memory requirements by quantifying the amount of entanglement in relation to the factored semiprime and other parameters. However, through our optimisations, we introduce an even more efficient method, allowing us to simulate a 60 qubit instance of Shor's algorithm with parameters that would be considered infeasible by the method of [24]. This stands as potentially the largest simulation of a non-trivial quantum circuit ever performed.

3.1 Review of Shor's algorithm

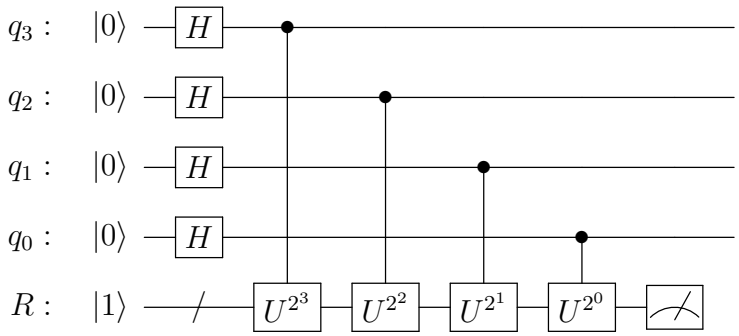
Shor's algorithm [2] can be used to factor a semiprime integer $N = p \times q$ of l binary digits in polynomial time with respect to l on a quantum computer. Presently, there is no known algorithm to perform this factorisation efficiently (i.e. in polynomial time) on a classical computer. The presumed difficulty of factoring very large semiprime numbers is the foundation beneath the security of the RSA public key cryptosystem [9] used to secure online communications [43], thus providing the motivation to build quantum computers capable of performing large enough instances of Shor's algorithm and to develop public key cryptosystems resistant to quantum computers [10]. For this review, we perform the circuit shown in Fig. 3.1.

Given N, a semiprime integer represented by at least l binary digits, a high-level circuit for Shor's algorithm as detailed in [39] consists of a 'lower' register R of l qubits initialised to the state $|1\rangle$ and an 'upper' register of $2l$ qubits initialised to $|0\rangle$:

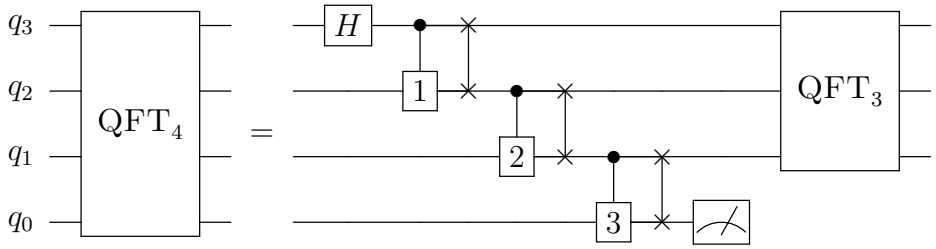
$$|\psi_{\text{init}}\rangle \leftarrow |0\rangle\,|1\rangle\,.$$

The Hadamard gate, Eq. (2.3), is then applied to each qubit of the upper register, creating the superposition

$$\begin{aligned}|\psi_{\text{superpos}}\rangle &\leftarrow (H^{\otimes 2l}) \otimes (I^{\otimes l})\,|\psi_{\text{init}}\rangle \\ &= \sum_{i=0}^{2^{2l}-1} |i\rangle\,|1\rangle\,,\end{aligned}$$



(a) Application of the controlled exponentiated U gates and measurement of the lower register



(b) The LNN quantum Fourier transform on the 4 upper register qubits, with interleaved measurement. The phase gate labelled by n performs $|0\rangle \to |0\rangle$ and $|1\rangle \to \exp(-i\pi/2^n)\,|1\rangle$.

Figure 3.1: The two main stages of Shor's algorithm. Shown for $l = 2$, qubits q_i form the upper register of $2l$ qubits and bundle R forms the lower register of l qubits. q_0 is the least significant bit of the upper register.

where we shall ignore normalisation constants in this explanation for clarity. Choosing a random integer a such that $1 < a < N$, exponents of the unitary operator

$$U: \quad U\,|x\rangle = |ax \bmod N\rangle \tag{3.1}$$

are applied to the lower register, controlled by qubits in the top:

$$\begin{aligned} |\psi_{\text{modexp}}\rangle \leftarrow & \sum_{i=0}^{2^{2l}-1} |i\rangle\, U^i\, |1\rangle \\ & = \sum_{i=0}^{2^{2l}-1} |i\rangle\, \left|a^i \bmod N\right\rangle \\ & = \sum_{\substack{j=0 \\ kr+j<2^{2l}}}^{r-1} \sum_{k<2^{2l}/r} |kr+j\rangle\, \left|a^j \bmod N\right\rangle, \end{aligned} \tag{3.2}$$

where r is the period of U. To determine this period r, the lower register is measured, forcing a choice of the index j in Eq. (3.2). This collapses the entanglement between the two registers, and we can now separate them. The upper register is now in state

$$|\psi_{\text{upper}}\rangle \leftarrow \sum_{\substack{k=0 \\ k<2^{2l}/r}} |kr+j\rangle \tag{3.3}$$

for the $0 \le j < r$ corresponding to the value $(a^j \bmod N)$ measured from the lower register in

Eq. (3.2). The circuit up to this point is shown in Fig. 3.1a.

The quantum Fourier transform (QFT), whose circuit is displayed in Fig. 3.1b, is then applied to this upper register:

$$\begin{aligned}\mathrm{QFT}\,|\psi_{\text{upper}}\rangle &= \sum_{s=0}^{2^{2l}-1} \sum_{\substack{k=0 \\ k<2^{2l}/r}} e^{2\pi i(kr+j)s/2^{2l}}\,|s\rangle \\ &= \sum_{s=0}^{2^{2l}-1} e^{2\pi ijs/2^{2l}} \sum_{\substack{k=0 \\ k<2^{2l}/r}} e^{2\pi ikrs/2^{2l}}\,|s\rangle\,.\end{aligned}$$

The upper register is then measured to produce a value s with probability proportional to

$$\left| \sum_{\substack{k=0 \\ k<2^{2l}/r}} e^{2\pi ikrs/2^{2l}} \right|^2 .$$

This implies that values of s such that $rs/2^{2l}$ is close to an integer are more likely to be measured, and concludes the quantum processing component of Shor's algorithm. s can be classically processed from here to produce r with high probability, by using the continued fractions algorithm on $s/2^{2l}$, and if r is even, the Euclidean algorithm is used to find the greatest common divisor between $a^{r/2} \pm 1$ and N, producing factors p and q of N [39].

3.2 Entanglement in Shor's algorithm

Since the space usage of an MPS scales with the amount of entanglement in the state, we should first examine the entangling properties of Shor's algorithm in order to determine an efficient embedding of the circuit's $3l$ qubits into an MPS with inherently linear connectivity. Eq. (3.3) for the state $|\psi_{\text{upper}}\rangle$ of the upper register following measurement of the lower register R tells us that $|\psi_{\text{upper}}\rangle$ is an evenly weighted superposition of computational basis states separated by r, the period of unitaries U from Eq. (3.1):

$$|\psi_{\text{upper}}\rangle = \sum_{\substack{k=0 \\ k<2^{2l}/r}} |kr+j\rangle\,.$$

If r can be factored such that $r = \beta \times 2^\alpha$ where β is odd, the last α bits in the binary representation of each computational basis element in the superposition $|\psi_{\text{upper}}\rangle$ must be identical. That is, measurement of R fully determines the least significant α qubits of the upper register. Therefore, immediately prior to the measurement of R, these α least significant qubits of the upper register only exhibit entanglement with R, and not between themselves or other qubits of the upper register. The entanglement related to the factor β of r is spread across the remaining $2l - \alpha$ most significant qubits of the upper register and the l-qubit lower register R immediately prior to R's measurement, since it cannot be localised to individual qubits by virtue of β's parity (odd). Following the measurement of the lower register, the entanglement between the α least significant qubits of the upper register with the lower register is collapsed, leaving these α qubits completely separable. The entanglement between the remaining $2l - \alpha$ qubits of the upper register with the lower register is also collapsed, leaving these $2l - \alpha$ qubits entangled only with each other.

This suggests a further partitioning of the upper register. From now on, we will refer to the set of α least significant qubits in the upper register as A, and the remaining $2l - \alpha$ most significant qubits of the upper register as B.

3.3 MPS simulation approach

3.3.1 Previous approaches

A high-level distributed memory simulation of Shor's algorithm was performed by Wang, Hill and Hollenberg in [24]. To briefly summarise their method, the lower register R of Shor's algorithm was kept contracted as a sparse MPS tensor for a 2^l dimensional qudit, only storing tensor indices corresponding to the values currently occupied in the lower register as the controlled exponentiated U gates are sequentially applied (Fig. 3.1a) to produce Eq. (3.2). With R initialised to $|1\rangle$, each of the $2l$ upper register qubits q_i for $0 \leq i < 2l$ is entangled with R by applying the following procedure to each q_i in order of descending i:

- An initially separable qubit q_i is inserted between the previous upper register qubit and R by altering the bond sizes appropriately and is initialised to the state $(|0\rangle + |1\rangle)/\sqrt{2}$ to represent the initial round of Hadamard gates.
- The gate U^{2^i}, which may be formed by repeated squaring of U in Eq. (3.1), is applied to R and controlled by q_i. This involves contracting q_i with R beforehand.
- This contraction is then decomposed back into qubit q_i and the lower register qudit R by using the trivial decomposition Eq. (2.1). By Eq. (3.2), we do not require rank minimisation through a rank revealing decomposition at this stage, so the apparent rank from the trivial decomposition is equal to the Schmidt rank.

The authors of [24] note that these steps may be combined into a single operation for a qubit q_i. In our simulation, we were able to replicate this operation through sufficient bookkeeping of indices. The effect of these controlled operations between each of the $2l$ upper register qubits with the lower register R is to quickly initialise an MPS that encodes the state $|\psi_{\text{modexp}}\rangle$ in Eq. (3.2). Since all upper register qubits are on one side of the lower register qudit, applying these controlled gates ensures that the Schmidt ranks at each bipartition are non-decreasing toward the lower register qudit, culminating in a rank of r from Eq. (3.2) between the least significant qubit q_0 of the upper register with the lower register. Since R's tensor only stores indices for occupied values, it is treated as an r-dimensional qudit at this point, also from Eq. (3.2).

This MPS is not in canonical form, since the trivial decomposition was used instead of the SVD. As such, the authors perform measurement of the lower register by directly calculating R's reduced density matrix through a tensor network contracting involving all upper register qubits, in a similar fashion to Fig. 2.7b. R is then measured, and entanglement is collapsed to reduce space usage by performing a sweep outward from R using the SVD. Since the lower register is now separable at this stage, it is removed from the MPS and the remaining upper register of $2l$ qubits encodes the state $|\psi_{\text{upper}}\rangle$ in Eq. (3.3).

Puzzlingly, the quantum Fourier transform (QFT) is performed in [24] with non-LNN gates, sequentially contracting qubits together and eventually resulting in a 2^{2l} dimensional state vector of the upper register. An LNN circuit for the QFT does exist [44], as shown in Fig. 3.1b, and is more suitable for MPS simulation since contraction into a state vector ultimately limits the size of the simulation. However, the full contraction approach does perform exceptionally well timewise due to parallelism and a lack of decomposition. Though in that case, since the QFT is numerically identical to the discrete Fourier transform (DFT) on the elements of a

quantum state vector, the authors could instead have used a distributed DFT library such as FFTW [45] on the upper register's state vector for greater performance if they were willing to accept the memory costs of contracting to a state vector, since this is a high-level simulation after all.

It is also worth noting that this state vector for the final state of the upper register depends on the result measured in the lower register, and may differ based on this result. By the principle of implicit measurement [39, section 4.4], the final upper register measurement statistics of Shor's algorithm should not depend on a measurement of the lower register: we describe it for clarity and use it to collapse entanglement in our MPS. Therefore, providing the full state vector of the combined lower and upper registers, or the reduced density matrix of the upper register (or perhaps even just its diagonal elements, corresponding to the upper register's measurement probabilities) should increase the consistency of their method.

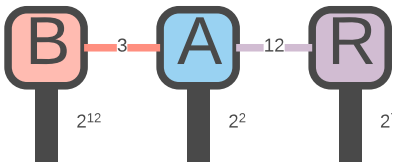
3.3.2 Simulation through QCMPS

To avoid entirely contracting an MPS to return a state vector which would negate any space savings, we instead decided to simulate Shor's algorithm as a sampling problem. That is, when Shor's algorithm is run on actual quantum computing hardware, only measurement operations allow us to extract information from the quantum state. The results of these measurements are distributed according to some discrete probability distribution dictated by the quantum circuit itself. Instead of requiring an entire state vector from our simulation, like [24] provides, we instead only wish to perform measurement operations whose results sample from the underlying distribution. In this way, our simulation mimics the process of a quantum computer. Performing simulations in this way is ideal for MPS, since it mitigates the need to contract to a state vector, and the entanglement collapse following a measurement reduces memory usage.

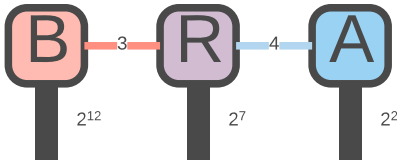
From the description of Shor's algorithm in section 3.1, the coefficients in all quantum states prior to the QFT are real. Therefore, we can approximately halve our time and space requirements for this section by storing our MPS elements as real scalars instead of complex scalars of equal precision per [24], and then converting to complex scalars prior to the QFT. These savings come at an opportune time due to the amount of entanglement before measurement of the lower register R.

The centrepiece of our additional optimisations, however, is to make the lower register qudit R, well, the centrepiece of our MPS. When considering the entangling properties of Shor's algorithm as detailed in section 3.2, if A is the set of α least significant qubits of the upper register, where $r = \beta \times 2^\alpha$, and B consists of the remaining qubits in the upper register, then the approach of [24] is to initialise an MPS in the order $B{:}A{:}R$. This has a significant space disadvantage, since A is not entangled to B, but the entanglement between B and R crosses through A and multiplies the Schmidt ranks within A by β. Instead, we have chosen to perform our simulations by positioning our partitions as $B{:}R{:}A$ in our MPS, which allows direct connectivity, representing the multipartite entangling structure of Shor's algorithm, between R with A and R with B. With this layout, each of the α qubits in A stores β^{-2} as many elements in its tensor when compared with [24] (a factor of β^{-1} each for the left and right bonds). Therefore, we can better classify the difficulties of simulating Shor's algorithm through not just r as believed in [24], but also through the factors of r. A schematic example is shown in Fig. 3.2.

Since our simulation should not depend on knowing the values r, α or β beforehand, we must be able to dynamically detect the transition from qubits in B to qubits in A when applying the controlled exponentiated U gates. In section 3.2, we noted that there was entanglement between the qubits in B and R, and also amongst the qubits in B when all controlled exponentiated U gates have been applied. Therefore, as more and more controlled U gates are operated on qubits



(a) The MPS qudit layout of [24]



(b) Our MPS qudit layout, showing lower space requirements for A

Figure 3.2: A comparison of the two MPS layouts of qudit groups B, A and R. In this example, $l = 7$, $N = 65$ and $a = 2$, producing $r = 12 = 3 \times 2^2$. A consists of $\alpha = 2$ qubits, B consists of the remaining $2l - 2 = 12$ qubits, and R is the lower register of $l = 7$ qubits. Index dimensions are indicated, and bond tensors are omitted due to the use of the trivial decomposition.

in order of descending significance, the Schmidt ranks with R will temporarily stop increasing (at a value of β) when this amount of entanglement has been reached. Only when we start operating on qubits in A does the entanglement to R begin to increase again. By detecting this plateau and subsequent rise in the Schmidt ranks, we can detect the boundary between A and B and begin to place qubits from A on the opposite side of R.

Local MPS measurement of the lower register requires it to be in canonical form. When sufficiently many controlled exponentiated U gates have been operated to initialise the systems B:R, we perform a right sweep to left canonicalise the qubits of B before the qudit tensor of R increases in size due to entanglement with A. As each of the α qubits in A interacts with R, the Schmidt ranks must sequentially double to reach a lower register occupancy of $r = \beta \times 2^\alpha$ given the contribution of β from B. Use of the trivial decomposition Eq. (2.1) therefore ensures that the tensors of each qubit in A are reshapes of the identity matrix. Thus, right canonicalising them is simply a matter of normalising them correctly. R is then measured to a one-dimensional qudit per Eq. (2.7) then contracted with the smaller of its adjacent qubits to remove it. Sweeps are then performed outward from this site to collapse entanglement with R and canonicalise the entire MPS consisting now of systems B:A. Since the only remaining entanglement exists between the qubits of B, the highest Schmidt rank of the MPS at this stage is β, and the qubits of A are completely separable.

We then apply the LNN circuit for the QFT, shown in Fig. 3.1b, to our remaining qubits. Since each controlled phase gate is immediately followed by a swap on the same qubits, we apply these as a combined operation. We note that the structure of the LNN QFT circuit allows us to measure any qubits that no longer require further operations. By performing these measurements as soon as possible and recanonicalising afterwards, we help mitigate any extra entanglement introduced during the progression of the QFT circuit.

3.4 Benchmarks

QCMPS has the option of specifying whether to use the slower but more space efficient SVD algorithm based on QR iteration [31] implemented in ScaLAPACK [30] as used in [24], or the faster divide and conquer algorithm [33] unique to Elemental [32], ontop of which QCMPS is built. We chose to run benchmarks using the divide and conquer approach for SVD in order to test this unique feature of QCMPS, and we use Elemental's element-wise cyclic distribution of tensor elements amongst the multiple processes in the process grid.

We began by running the same instances benchmarked in [24] for their MPI results, also at double precision. These were run on a single Intel Xeon Processor E5-2683 v4 with a base core clock of 2.10 GHz, using the Intel Parallel Studio XE suite for compilers and intraprocess BLAS and LAPACK [34] and Open MPI [46] for our MPI implementation. Despite having a lower frequency than the unspecified AMD Opteron processors at 2.5 GHz used in [24], timing results are still not directly comparable: factors such as compiler optimisation, instructions per cycle (IPC) performance, cache size, available instruction sets and communication bandwidth and latency obscure true performance differences between implementations. Furthermore, we limited the space usage for these results to 8 GB of RAM to showcase our optimisations against the 16 GB required in [24]. That being said, our implementation appears to result in a significant overall performance increase (table 3.1).

l	r	α	β	n_{proc}	t_U	t_{meas}	t_{QFT}	t_{total}
13	3870	1	1935	4	118	24	288	420
				16	48	10	105	163
14	8036	2	2009	4	146	38	585	769
				16	58	15	205	278
15	16104	3	2013	4	157	48	970	1175
				16	63	19	339	421

Table 3.1: QCMPS benchmarks for simulating Shor's algorithm with our optimisations, with times for the exponentiated U gates, measurement and QFT listed in seconds. With n_{proc} cores, we simulated the three cases $l = 13$, $N = 8189$, $a = 10$; $l = 14$, $N = 16351$, $a = 2$ and also $l = 15$, $N = 32663$, $a = 6$.

The parameters chosen in [24] produce values of r close to its upper bound of $N/2$ [39], since this was how they classified their difficulty in simulating Shor's algorithm. Notably, [24] shows a significant increase for the total time taken as r is increased. Our approach classifies difficulty based on the factors of r, especially the odd factor β of r, which are similar between these three cases. This is reflected in our results by the similarity in the times required, in that the time required for a value of l does not quite double like in [24] when l is incremented. We also note that [24] did not provide any results for circuits larger than $l = 15$. We suspect that contracting the upper register of an $l \geq 16$ case instance produces a state vector of size greater than 2^{32}, and since the reference implementation of ScaLAPACK [30] usually indexes elements through 32 bit integers, the code from [24] might have just failed to produce correct results for $l \geq 16$. However, without access to their source code, this remains pure speculation. QCMPS is subject to a greater limit due to Elemental's use of 64 bit integers for indexing.

Furthermore, our individual timings for the application of the exponentiated U gates and the QFT appear slower than the results of [24], but this is due to our t_U being used to record the time to perform recanonicalisation and us choosing to use the LNN QFT with interleaved measurements respectively. Our time saved in measurement more than makes up for this. Additionally, whilst [24] is able to claim $1/n_{\text{proc}}$ time scaling in his results due to the parallelism of MPS contraction, we relied more on MPS decomposition through the SVD to keep our memory requirements low. It would appear then that the SVD scales differently with n_{proc} since we do not observe $1/n_{\text{proc}}$ time scaling.

To benchmark a truly distributed memory implementation, we also performed some benchmarks across multiple nodes on Magnus [25], a Cray XC40 supercomputer with 24 cores at 2.60 GHz and 64 GB of RAM per node. The results of table 3.2 were run using the GNU Compiler Collection suite for compilers, OpenBLAS [35] for intranodal BLAS and LAPACK and Cray's proprietary MPI implementation. To especially distinguish our approach from that of

[24], we chose the parameters $N = 961307$ and $a = 5$ for our flagship $l = 20$ benchmark. This has a period $r = 479568 = 29973 \times 2^4$. Since r here is extremely high, it would be infeasible to simulate this in a similar fashion to [24], even if final contraction of the MPS was not performed. In our simulation, the $\alpha = 4$ qubits in A significantly decrease the amount of resources required, just from an understanding of the entangling properties of Shor's algorithm. As such, we were able to perform a high-level simulation of sampling a measurement from Shor's algorithm on 60 qubits using a total of 216 nodes, 5184 cores and 13.824 TB of memory within 8 h. This is possibly the current largest high-level simulation of Shor's algorithm or perhaps any other non-trivial quantum algorithm.

l	r	α	β	n_{node}	t_U	t_{meas}	t_{QFT}	t_{total}
16	28140	2	7035	2	1538	353	4290	6181
17	57516	2	14379	24	1694	406	4544	6644
20	479568	4	29973	216	4271	1496	20236	26003

Table 3.2: Further QCMPS benchmarks, this time across multiple nodes of a supercomputer. Each node has 24 cores and 64 GB of RAM. With n_{node} nodes, we simulated the three cases $l = 16$, $N = 56759$, $a = 2$; $l = 17$, $N = 124631$, $a = 2$; and also $l = 20$, $N = 961307$, $a = 5$.

3.5 Future simulations

Since this is only a high-level simulation of Shor's algorithm, the next step is to perform a simulation of a circuit for Shor's algorithm at the gate level. Several gate-level decompositions of Shor's algorithm already exist [47–49], but we would have to insert our own swap gates to map these circuits to an MPS. Instead, by using an LNN circuit for the entirety of the quantum processing component of Shor's algorithm (and not just for the QFT as we have in our simulations) such as the circuit proposed by Fowler, Devitt and Hollenberg in [44], we can make direct use of QCMPS library routines, including its linear nearest neighbour generalised binarily controlled gates (section 2.5), to perform a low-level simulation of Shor's algorithm. The circuit of [44] notably requires only $2l+4$ qubits to factor a semiprime of l bits, compared to the $3l$ of our simulation. A smaller quantum system, coupled with the intermediate measurements made throughout the course of their circuit might lead to lower space usage than our approach. However, we expect the $8l^4 + 40l^3 + 58l^2 + 2l - 2$ required gates to heavily increase the time usage. With an LNN circuit for Shor's algorithm, we can also apply the matrix product density operator (MPDO) functionality of our library to simulate the circuit's susceptibility toward noise and errors.

Additionally, we might also wish to determine if similar techniques for manipulating the multipartite entanglement in Shor's algorithm for semiprime factorisation may also be applied to the discrete logarithm problem [2] on which elliptic curve cryptograph [50] is based.

Chapter 4

Google Supremacy Circuit Simulations

In the last chapter, we investigated MPS methods for simulating Shor's algorithm [2]. This algorithm allows a quantum computer to factor a semiprime integer in polynomial time with respect to its length in digits, a feat not demonstrated on a classical computer. Algorithms that show an apparent advantage of quantum computation over classical computation allude to an inherent 'quantum supremacy' over classical models of computation (of which the Turing machine [1] is one).

This notion of quantum supremacy has lead to the development of problems to specifically pinpoint instance sizes efficiently solvable on an ideal quantum computer but infeasible on any reasonable amount of classical computational hardware. There are several theoretical and experimental challenges in demonstrating quantum supremacy, however. Open questions in computer science regarding complexity theory (of which P versus NP [51] is a famous instance) currently prevent us from making concrete claims of any theoretical quantum supremacy: for example, if an efficient classical algorithm for integer factorisation did exist, then performing a sufficiently large instance of Shor's algorithm on a quantum computer would not demonstrate supremacy. Implementing quantum hardware resistant enough from decoherence and noise effects to perform large enough instances to demonstrate supremacy is also experimentally challenging.

Further examples [13] of problems for demonstrating quantum supremacy include boson sampling [52], where the aim is to accurately sample the outputs of a linear optics network with single photon inputs. Through combinatorial arguments [14], the supremacy point of a linear optics quantum computer is expected to be around 50 single photon inputs for boson sampling.

In this chapter, we examine the problem of sampling from random quantum circuits with a structure proposed by Google's Boixo and Isakov et al. in [17]. Henceforth, we shall refer to this problem as the Google supremacy problem. Specifically, we wish to determine if MPS techniques can be applied for classical simulation of similar circuits to refine this problem's supremacy point.

4.1 Introduction to the Google supremacy problem

Google's supremacy problem [17] involves sampling measurements from the results of specific randomly generated circuits on a two-dimensional nearest-neighbour architecture. If a random circuit U operates $|\psi\rangle \leftarrow U|\psi\rangle$ on an n qubit system (or equivalently, a single $N \equiv 2^n$ level system), the problem for either a classical computer or experimental implementation of a quantum computer is to sample bit strings $x \in \{0,1\}^n$ in, say, the computational basis according to the probability distribution resulting from applying U.

The random two-dimensional circuits are generated according to a basic structure defined

in [17, section IV]. To briefly review, n qubits are arranged in a rectangular grid of dimensions $p \times q$ and the state is initialised to $|0\rangle^{\otimes n}$. The Hadamard gate H of Eq. (2.3) is applied to each qubit in the zeroth clock cycle. The other gates used in this circuit have the following matrix representations in the computational basis:

$$\mathrm{C}Z = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}, \quad \sqrt{X} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -i \\ -i & 1 \end{bmatrix}, \quad \sqrt{Y} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}, \quad T = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}. \tag{4.1}$$

For each following clock cycle,

- a configuration of nearest neighbour controlled Z ($\mathrm{C}Z$) gates is applied from a cycle of eight possible configurations [17, Fig. 6], and
- for a qubit q that was operated on by a $\mathrm{C}Z$ gate of the last clock cycle but not of this current clock cycle, the T gate is applied to q if q's previous single qubit gate was the initial Hadamard, else a random gate sampled uniformly from $\left\{\sqrt{X}, \sqrt{Y}, T\right\}$ that is different from the last single qubit applied to q is applied.

An example for the beginning of a random circuit in one-dimension is given in Fig. 4.1, where there are three $\mathrm{C}Z$ configurations to cycle through instead of the eight for the two-dimensional problem.

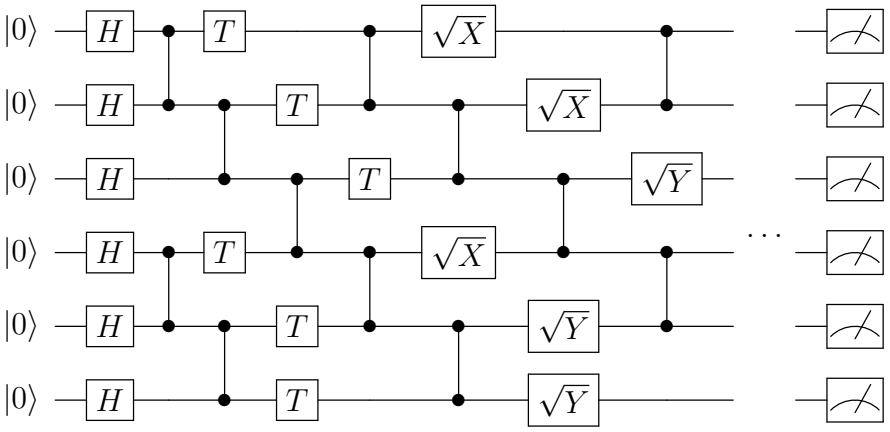


Figure 4.1: An example circuit for the Google supremacy problem in one dimension. Note the three cycling configurations of $\mathrm{C}Z$ gates instead of the eight for the two-dimensional case.

If the set of probabilities $p_U(x) \equiv \Pr(X_U = x) = |\langle x|\psi\rangle|^2$ of measuring bit string $x \in \{0,1\}^n$ for discrete random variable X_U is produced by circuit U, U can be performed m times to produce a sample $S = \{x_1, \ldots, x_m\}$ of measured bit strings x_j with probability $\Pr_U(S) = \prod_{x_j \in S} p_U(x_j)$. By rearranging, the central limit theorem implies that

$$-\frac{1}{m} \log(\Pr_U(S)) \to \mathrm{H}(X_U) \tag{4.2}$$

as the sample size m is increased, where

$$\mathrm{H}(X_U) = -\sum_x p_U(x) \log(p_U(x))$$

is the entropy of X_U.

As the number of clock cycles of circuit U increases, it is argued with accompanying numerical evidence in [17] that the probabilities $p_U(x)$ are independent and identically distributed with probability density function converging to the Porter-Thomas distribution

$$\Pr(a \leq p_U(x) \leq b) = \int_a^b Ne^{-Np}\,\mathrm{d}p\,, \tag{4.3}$$

where $N = 2^n$ is the number of possible bit strings measurable from n qubits. In this case, a change of variable yields

$$\begin{aligned} \mathrm{H}(X_U) =& -N\int_0^\infty p\log(p)Ne^{-Np}\,\mathrm{d}p \\ =& \log(N) - 1 + \gamma, \end{aligned} \tag{4.4}$$

where γ is the Euler-Mascheroni constant.

Consider an algorithm A that takes a specification of the (gates of) circuit U and produces a sample $S_A = \{x_1^A, \ldots, x_m^A\}$ of resulting bit strings, where each result is produced with probability $p_{A(U)}(x) \equiv \Pr(X_{A(U)} = x)$ for random variable $X_{A(U)}$. The probability $\Pr_U(S_A)$ of actually observing this sample from circuit U has a similar expression to Eq. (4.2):

$$-\frac{1}{m}\log(\Pr_U(S_A)) \to \mathrm{H}(X_{A(U)}, X_U),$$

in the large m limit by the central limit theorem, where

$$\mathrm{H}(X_{A(U)}, X_U) = -\sum_x p_{A(U)}(x)\log(p_U(x))$$

is the cross entropy of $X_{A(U)}$ with respect to X_U. Note that if algorithm A is to just sample from U itself, then this reduces to Eq. (4.2).

Instead, if an algorithm randomly samples bit strings uniformly such that $p_0(x) \equiv \Pr(X_0 = x) = 1/N$ for random variable X_0, we have

$$\begin{aligned} \mathrm{H}_0 \equiv\ & \mathrm{H}(X_0, X_U) \\ =& -\sum_x \frac{1}{N}\log(p_U(x)) \\ =& -N\int_0^\infty \frac{1}{N}\log(p)Ne^{-Np}\,\mathrm{d}p \\ =& \log(N) + \gamma. \end{aligned} \tag{4.5}$$

This is used to define the cross entropy difference, a measure of how effectively an algorithm A can sample from the distribution of circuit U when compared to a uniform sampler:

$$\Delta\mathrm{H}(X_{A(U)}) \equiv \mathrm{H}_0 - \mathrm{H}(X_{A(U)}, X_U). \tag{4.6}$$

From this, the fidelity α at which A can sample from a general circuit U of depth d is given by the ensemble average of the cross entropy difference across all circuits of depth d:

$$\alpha = \mathbf{E}_U\big[\Delta\mathrm{H}(X_{A(U)})\big]. \tag{4.7}$$

From Eqs. (4.4) and (4.5), $\alpha = 1$ when an ideal sampler is considered, so in general, $0 \leq \alpha \leq 1$.

The Google problem defines that quantum supremacy is achieved by a physical quantum

computer that can achieve a higher fidelity than C, the highest fidelity attainable by a classically performed algorithm A. In [17, section V], computational complexity arguments are made to support the claim that high fidelity can only be classically achieved via a direct simulation of circuits U. Supercomputers currently exist [15] with enough memory to store the state vector of $n = 48$ qubits in single precision, requiring 4 PiB of memory. Hence, quantum supremacy is unattainable below this number of qubits, neglecting error in floating-point arithmetic. For larger systems, the depth d to reach the Porter-Thomas distribution, Eq. (4.3), cripples the lower bound of C for polynomially sized samples of Feynman paths [17, appendix H].

Our aim for the rest of this chapter is to investigate whether or not matrix product techniques can be applied to raise this lower bound for C.

4.2 One-Dimensional Circuit

The original Google problem defines the random circuits using nearest neighbour CZ gates on a two-dimensional array of qubits. However, the rules for generating these random circuits easily generalise to an LNN version, ideal for simulation through QCMPS. Rather than simulating the original two-dimensional circuits on an MPS or MPDO, we have chosen to simulate the one-dimensional versions instead since SWAP gates are not required, but also to investigate their convergence and noise properties in comparison to the results presented by [17]. The one-dimensional circuits cycle through three configurations of nearest neighbour CZ gates as demonstrated in Fig. 4.1, resulting from the claim in [17] that two distinct CZ gates cannot currently be operated simultaneously on adjacent qubits.

When simulated on an MPS, the output distributions of these circuits may be sampled by sequentially measuring each qubit with recanonicalisation, avoiding the need to fully contract the MPS into a state vector or the MPDO into a density matrix.

4.2.1 Porter-Thomas Convergence

As the depth d of circuit U on n qubits increases, the probability $p_U(x)$ of measuring a bit string x follows a probability density function converging to the Porter-Thomas distribution, Eq. (4.3). The claim in [17, section IIA] is that this required depth for convergence is proportional to $n^{1/D}$, where D is the dimensionality of the qubit array. Therefore, we expect the depth of convergence for a one-dimensional circuit to eventually exceed that of a two-dimensional circuit with an equal number of qubits.

We numerically simulated instances of the one-dimensional Google problem to determine when we could expect this convergence. From Fig. 4.2, we observe that convergence occurs approximately at least at depth $d = 100$ for $n = 20$ qubits, a much larger requirement than Google's analysis of the two-dimensional circuits of depths less than 40 on much larger systems [17, Fig. 9]. Additionally, since we expect the output states to be maximally entangled, the Schmidt ranks should saturate when convergence has been achieved. From Fig. 4.3, this only occurs after a depth of 125 of 20 qubit circuits, again showing the larger circuit requirements for the one-dimensional problem.

4.3 MPS Truncation

Truncation of bonds within an MPS, described in subsection 2.4.4 as a lossy compression scheme for an MPS, allows us to approximate a quantum state with a less entangled state, and therefore trades greater precision for a lower space requirement. If χ is the largest Schmidt rank of a bipartition (i.e. the largest bond dimension) of an MPS of n qubits and χ is bounded

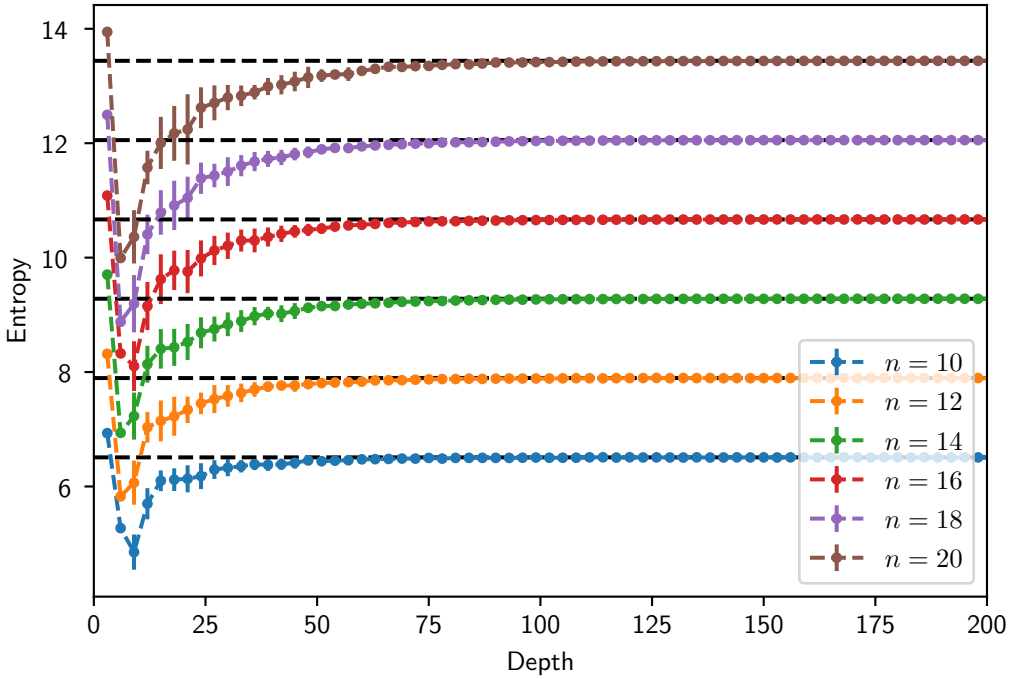


Figure 4.2: Mean output distribution entropies as a function of circuit depth. Sampled over 100 random circuits each, the error bars represent one standard deviation. The black dashed lines are the respective Porter-Thomas entropies for the number of qubits n.

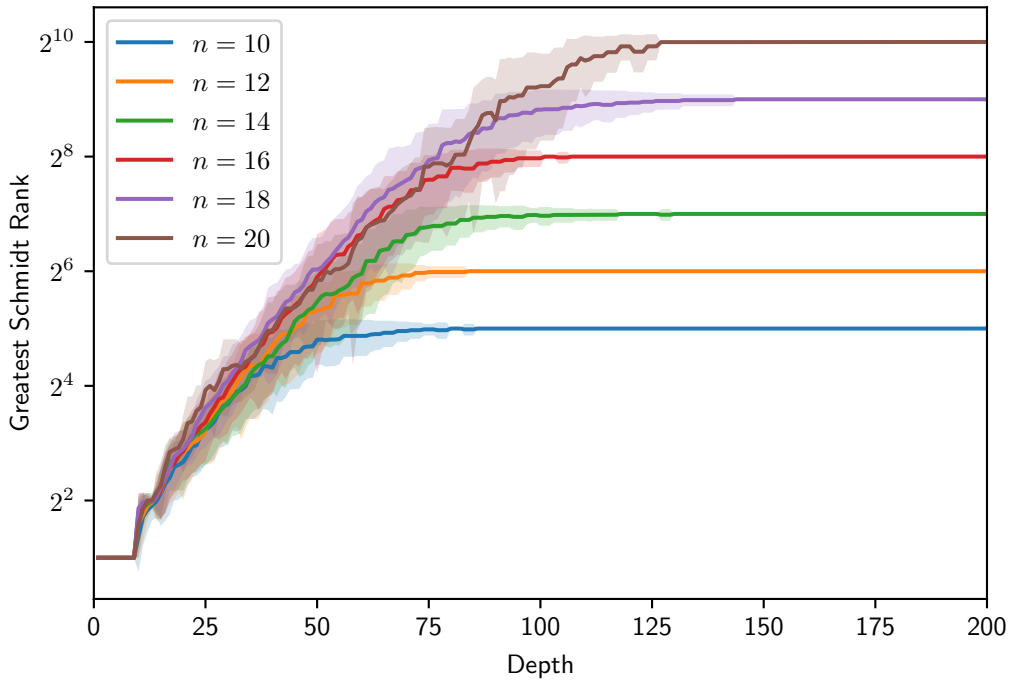


Figure 4.3: Mean maximum Schmidt ranks in the MPS as a function of depth. Sampled over 100 instances, the shading represents one standard deviation.

polynomially in n, then the space requirements of such an MPS also scale polynomially in n and gate operations can be simulated in polynomial time [21].

When the one-dimensional Google problem with sufficient depth for the measurement probabilities to converge to the Porter-Thomas distribution is performed, the resulting state should be highly entangled, producing an MPS without any space or time advantage over a state vector. Therefore, we perform truncation across all clock cycles by truncating every MPS bond following a CZ gate (the only multiple qubit gate used in the Google problem and hence the only one capable of raising the Schmidt rank of a bipartition) to some maximum Schmidt rank χ. Each of these truncations is performed without the accompanying recanonicalisation sweeps, but a full recanonicalisation with renormalisation is performed just prior to final measurement.

We may now investigate the fidelity associated with truncated MPS simulations of the one-dimensional Google problem. Fig. 4.4 shows the fidelity α of a 20 qubit circuit dropping with respect to the circuit depth, as we expect for truncated MPSs. If we consider the $\chi = 256$ case (a truncation factor of 0.25 for 20 qubits), corresponding to the maximum Schmidt rank achievable with an $n = 16$ circuit, it appears that a fidelity of $\alpha \approx 0.3$ may still be achieved if we consider Porter-Thomas convergence to have occurred at a depth of 150. If this trend continues to larger one-dimensional circuits, it might be possible to simulate up to four more qubits up to this depth than an untruncated MPS, with still very acceptable fidelity. However, we lack to numerical evidence to support this claim at the moment.

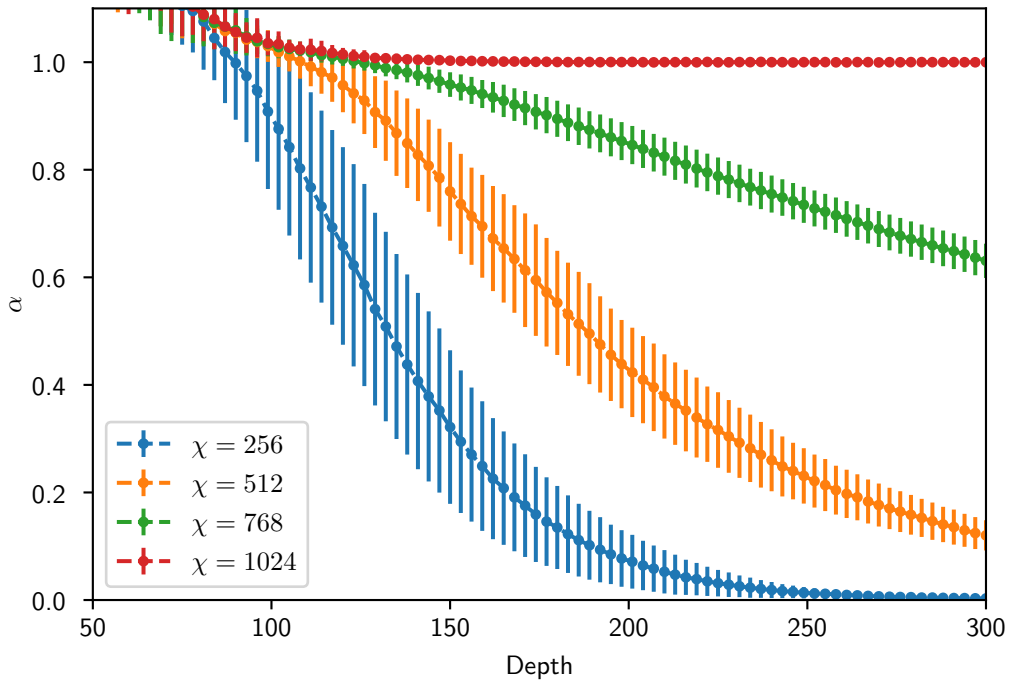


Figure 4.4: Mean fidelities α for $n = 20$ qubits at different truncation ranks χ. Note that α is only meaningful when convergence to the Porter-Thomas distribution has occured. Error bars represent one standard deviation.

4.4 MPDO Noise Channel Simulations

The fidelity α from Eq. (4.7) of some sampling algorithm to an ideal sampler may also describe experimental implementations of quantum hardware, subject to noise and errors during the

course of a circuit. In [17, section III], a discrete error model consisting of bit-flip, Eq. (2.12), and depolarising channels, Eq. (2.13), was considered in order to simulate the sampling results that experimental quantum hardware might produce. Specifically, a bit-flip channel is placed after each qubit immediately following initialisation of the state to $|0\rangle^{\otimes n}$ with probability r_{init} to simulate initialisation errors, and immediately preceding final measurement with probability r_{meas} to simulate measurement errors. A two-dimensional (four-dimensional) depolarising channel is applied following each one (two) qubit gate with probability $\frac{4}{3}r_1$ ($\frac{16}{15}r_2$) to simulate one (two) qubit errors. These normalisation prefactors of $\frac{4}{3}$ and $\frac{16}{15}$ occur in order to keep our description of the depolarising channel consistent with [17].

Since the fidelity α of some sampler A requires the cross entropy difference, Eq. (4.6), of A with an ideal sampler of U, calculating α becomes difficult when the number of qubits n is large, since the cross entropy difference requires access to the individual probabilities $p_U(x)$ of measuring each bit string x from circuit U. It is argued in [17, appendix A] with numerical evidence that the fidelity α can be easily related to the error rates via

$$\alpha \approx \exp(-r_1 g_1 - r_2 g_2 - r_{\text{init}} n - r_{\text{meas}} n), \tag{4.8}$$

where g_1 (g_2) is the total number of one (two) qubit gates. This allows us to estimate the cross entropy difference without access to each probability $p_U(x)$. To provide numerical evidence of Eq. (4.8) for the one-dimensional Google problem, we simulated this error model in Fig. 4.5 by averaging across many MPS simulations where bit-flips and depolarisations had their respective probabilities to occur at their respective locations.

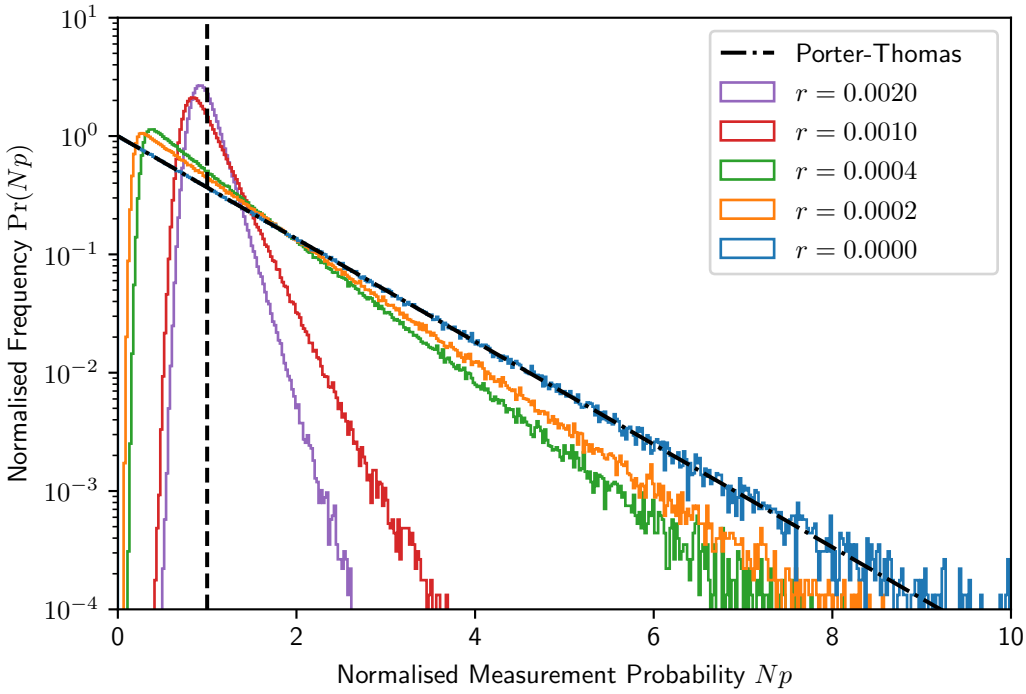


Figure 4.5: Output distributions on $n = 20$ qubits for different fidelites, corresponding to approximate fidelites $\alpha \in \{0.11, 0.34, 0.65, 0.8, 1\}$ in order of descending error rate, with $r_{\text{init}} = r_{\text{meas}} = r_2 = 10r_1 = r$.

We find good agreement with the ansatz output density matrix ρ resulting from this error

model, given by [17, Eq. (20)]

$$\rho = \alpha \left|\psi_d\right\rangle\!\left\langle\psi_d\right| + (1-\alpha)\frac{I_N}{N},$$

where ψ_d is the state resulting from circuit U without errors and the combined results of the errors is to depolarise $|\psi_d\rangle\langle\psi_d|$ with probability $(1-\alpha)$ (c.f. Eq. (2.13)). The slight differences in the widths and shapes of the distribution arise from the number samples we took, as well as residual correlations after the error operations [17, appendix A].

Classical simulation of this error model is possible using the MPDO formalism implemented in QCMPS. Given a set of error rates $r_{\{1,2,\text{init},\text{meas}\}}$ for the model, it is possible that the entanglement between qubits in the MPDO introduced by the CZ gates is also collapsed by the noise channels. Since the space requirements for a MPDO of maximal Schmidt rank, $\mathcal{O}(2^{2n})$, are much greater than that of an MPS of maximal Schmidt rank, $\mathcal{O}(2^n)$, only sufficiently large error rates can be classically simulated. For circuits with sufficient clock cycles to exhibit Porter-Thomas convergence, this may result in significant loss of fidelity, especially since scaling of the convergence depth in the one-dimensional Google problem is worse compared to the two-dimensional problem.

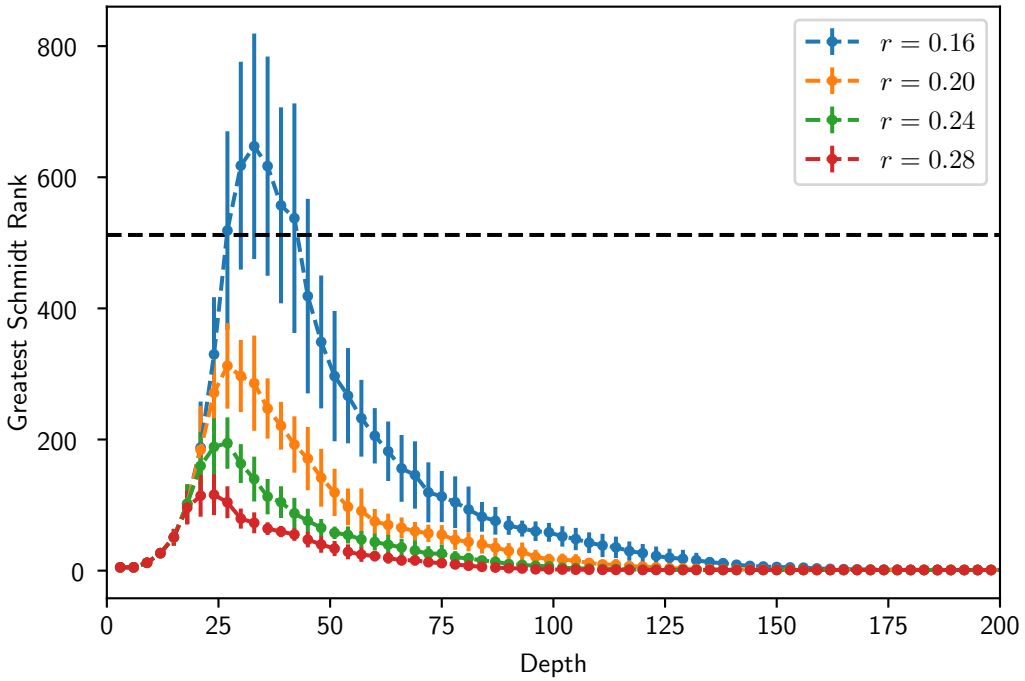


Figure 4.6: Schmidt ranks within a 20 qubit MPDO for error rates $r_{\text{init}} = r_{\text{meas}} = r_2 = 10r_1 = r$. Error bars represent one standard deviation, and the black bar represents the Schmidt rank where the MPDO's space usage matches an MPS also on 20 qubits.

Our results of Fig. 4.6 demonstrate the capacity for noise channels to collapse entanglement within an MPDO, but the added space usage of the density matrix formalism imposes a minimum set of error rates in order to scale to a larger number of qubits. Coupled with the increased depth of our one-dimensional circuits to achieve Porter-Thomas convergence, fidelity is greatly diminished. Therefore, we concede that the one-dimensional Google problem is resistant to our MPDO approach.

Chapter 5

Conclusion

To simulate the circuits of Shor's algorithm and the Google supremacy problem, we developed the highly optimised numerical library QCMPS for manipulating matrix product states and matrix product density operators. Conventionally, it is tailored for simulations of quantum circuits, but may readily be extended to perform simulations of other algorithms pertinent to condensed matter physics, such as the density matrix renormalisation group method and time-evolving block decimation. This library introduces several unique features, including more efficiently simulating specific linear nearest neighbour binarily controlled gates on potentially many sites of a pure state. It also uniquely implements the matrix product density operator formalism for simulating density matrices, allowing error models for the noise and decoherence effects experienced by a physical quantum machine to be simulated in a manner that takes advantage of the scaling in the simulation time and space requirements with respect to entanglement. Crucially, QCMPS was developed for distributed memory architectures, allowing it to scale to take advantage of the greater amounts of computational resources provided by supercomputers or computer clusters.

Building upon previous methods for simulating Shor's algorithm for semiprime factorisation, we described additional optimisations for our own implementation. Among these optimisations is the choice of simulating Shor's algorithm as a sampling problem, mimicking the process by which a real quantum computer might extract meaningful information from a quantum state produced by Shor's algorithm. As a sampling problem, the subsequent measurements made during the quantum Fourier transform serve to mitigate entanglement introduced by its gates, reducing rather than increasing the space requirements to perform this subroutine of Shor's algorithm.

By effectively characterising the multipartite entanglement over the course of Shor's algorithm and sensibly mapping it to a matrix product state of linear connectivity, we were able to refine previous estimates for the required simulation costs that were based on the period r to, instead, the factors of r. The culmination of all our optimisations combined is a successful simulation of a specifically chosen 60 qubit instance that previous estimations would have found extremely difficult space-wise, but one that we would ultimately perform with approximately 14 TB of memory over 8 h. Despite being a high-level simulation, this is still potentially the largest instance of a non-trivial quantum algorithm ever performed.

To extend our simulations of Shor's algorithm, it would be prudent to consider simulating a gate-wise decomposed circuit for Shor's algorithm, particularly decompositions into linear nearest neighbour gates that fit well with the linear connectivity of matrix product states. Doing so would also open the possibility of performing these simulations using the matrix product density operator functionality of QCMPS to examine the circuit's resilience to specific error models. Additionally, we might also wish to perform these simulations of Shor's algorithm when applied to the discrete logarithm problem as it appears in, say, elliptic curve cryptography.

We then turned our sights to Google's supremacy problem, which involves sampling the measurement results from the specification of a random circuit. Though the original circuits use nearest neighbour gates on 2-dimensional qubit arrays, we decided instead to approach the one-dimensional versions due to their suitability for our matrix product states and density operators.

Ultimately, we found that the amount of gates in a random one-dimensional circuit of the Google supremacy problem required for the measurement probabilities to converge to the Porter-Thomas distribution was significantly higher than 2-dimensional circuits over the same number of qubits. This increase in the number of gates results in the one-dimensional circuits being less resilient to errors described by the model given by Google, but also to matrix product state truncation. We were able to demonstrate this decay in sampling fidelity due to matrix product state truncation and, separately, Google's error model involving bit-flip and depolarising channels. Use of QCMPS' matrix product density operator functionality allowed us to observe the competing effects of the gates introducing entanglement and the depolarising channels collapsing it. For sufficiently high error rates, these circuits are viably simulated but result in poor sampling fidelity, affirming the one-dimensional Google problem's resistance to our matrix product methods.

Our goal from here is to extend the functionality of QCMPS by improving its interface and implementing better support for general matrix product operators. To achieve feature parity with other existing quantum simulators or tensor network libraries, we might also wish to implement support for general tensor networks, of which matrix product states and matrix product operators and density operators form a small subset, and also for increasingly popular heterogeneous GPU-accelerated computing architectures. As it stands, however, QCMPS should serve as a valuable tool for matrix product simulations of quantum algorithms.

Bibliography

[1]A. M. Turing, "On computable numbers, with an application to the entscheidungsproblem", Proceedings of the London Mathematical Society **s2-42**, 230–265 (1937).

[2]P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer", SIAM Journal on Computing **26**, 1484–1509 (1997).

[3]R. Jozsa and N. Linden, "On the role of entanglement in quantum-computational speed-up", Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences **459**, 2011–2032 (2003).

[4]C. H. Bennett, "Notes on landauer's principle, reversible computation, and maxwell's demon", Studies in History and Philosophy of Science Part B: Studies in History and Philosophy of Modern Physics **34**, Quantum Information and Computation, 501–510 (2003).

[5]A. C.-C. Yao, "Quantum circuit complexity", in Proceedings of 1993 ieee 34th annual foundations of computer science (Nov. 1993), pp. 352–361.

[6]E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser, "Quantum Computation by Adiabatic Evolution", eprint arXiv:quant-ph/0001106 (2000).

[7]L. K. Grover, "A fast quantum mechanical algorithm for database search", in Proceedings of the twenty-eighth annual acm symposium on theory of computing, STOC '96 (1996), pp. 212–219.

[8]J. Daemen and V. Rijmen, *The design of Rijndael: AES — the Advanced Encryption Standard* (Springer-Verlag, 2002), p. 238.

[9]R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems", Commun. ACM **21**, 120–126 (1978).

[10]D. J. Bernstein, "Introduction to post-quantum cryptography", in *Post-quantum cryptography*, edited by D. J. Bernstein, J. Buchmann, and E. Dahmen (Springer Berlin Heidelberg, Berlin, Heidelberg, 2009), pp. 1–14.

[11]A. Montanaro, "Quantum algorithms: an overview", npj Quantum Information **2**, 15023, 15023 (2016).

[12]A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, "Surface codes: towards practical large-scale quantum computation", Phys. Rev. A **86**, 032324 (2012).

[13]A. P. Lund, M. J. Bremner, and T. C. Ralph, "Quantum sampling problems, BosonSampling and quantum supremacy", npj Quantum Information **3**, 15, 15 (2017).

[14]B. T. Gard, K. R. Motes, J. P. Olson, P. P. Rohde, and J. P. Dowling, "An introduction to boson-sampling", in *From atomic to mesoscale* (WORLD SCIENTIFIC, 2015) Chap. Chapter 8, pp. 167–192.

[15]H. Meuer, E. Strohmaier, J. Dongarra, H. Simon, and M. Meuer, *Top500*, (Oct. 2017) `https://www.top500.org/`.

[16]M. J. Bremner, R. Jozsa, and D. J. Shepherd, "Classical simulation of commuting quantum computations implies collapse of the polynomial hierarchy", Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences **467**, 459–472 (2010).

[17]S. Boixo, S. V. Isakov, V. N. Smelyanskiy, R. Babbush, N. Ding, Z. Jiang, M. J. Bremner, J. M. Martinis, and H. Neven, "Characterizing Quantum Supremacy in Near-Term Devices", ArXiv e-prints (2016).

[18]J. Johansson, P. Nation, and F. Nori, "Qutip 2: a python framework for the dynamics of open quantum systems", Computer Physics Communications **184**, 1234–1240 (2013).

[19]M. Smelyanskiy, N. P. D. Sawaya, and A. Aspuru-Guzik, "qHiPSTER: The Quantum High Performance Software Testing Environment", ArXiv e-prints (2016).

[20]R. Orús, "A practical introduction to tensor networks: matrix product states and projected entangled pair states", Annals of Physics **349**, 117–158 (2014).

[21]G. Vidal, "Efficient classical simulation of slightly entangled quantum computations", Phys. Rev. Lett. **91**, 147902 (2003).

[22]U. Schollwöck, "The density-matrix renormalization group in the age of matrix product states", Annals of Physics **326**, 96–192 (2011).

[23]G. Vidal, "Classical Simulation of Infinite-Size Quantum Lattice Systems in One Spatial Dimension", Physical Review Letters **98**, 070201, 070201 (2007).

[24]D. S. Wang, C. D. Hill, and L. C. L. Hollenberg, "Simulations of shor's algorithm using matrix product states", Quantum Information Processing **16**, 176 (2017).

[25]The Pawsey Supercomputing Centre, *The pawsey supercomputing centre*, (Oct. 2017) `https://www.pawsey.org.au/`.

[26]S. van der Walt, S. C. Colbert, and G. Varoquaux, "The numpy array: a structure for efficient numerical computation", Computing in Science Engineering **13**, 22–30 (2011).

[27]M. P. Forum, *Mpi: a message-passing interface standard*, tech. rep. (Knoxville, TN, USA, 1994).

[28]E. Solomonik, D. Matthews, J. R. Hammond, J. F. Stanton, and J. Demmel, "A massively parallel tensor contraction framework for coupled-cluster computations", Journal of Parallel and Distributed Computing **74**, 3176–3190 (2014).

[29]J. A. Calvin and E. F. Valeev, *Tiledarray: a general-purpose scalable block-sparse tensor framework*, `https://github.com/valeevgroup/tiledarray`.

[30]L. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R. Whaley, "Scalapack: a linear algebra library for message-passing computers", in In siam conference on parallel processing (1997).

[31]G. Golub and C. Van Loan, *Matrix computations*, 4th ed., Johns Hopkins Studies in the Mathematical Sciences 3 (Johns Hopkins University Press, Dec. 2012).

[32]J. Poulson, B. Marker, R. A. van de Geijn, J. R. Hammond, and N. A. Romero, "Elemental: a new framework for distributed memory dense matrix computations", ACM Trans. Math. Softw. **39**, 13:1–13:24 (2013).

[33]E. R. Jessup and D. C. Sorensen, "A parallel algorithm for computing the singular value decomposition of a matrix", SIAM J. Matrix Anal. Appl. **15**, 530–548 (1994).

[34]E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, *LAPACK users' guide*, Third (Society for Industrial and Applied Mathematics, Philadelphia, PA, 1999).

[35]Q. Wang, X. Zhang, Y. Zhang, and Q. Yi, "Augem: automatically generate high performance dense linear algebra kernels on x86 cpus", in Proceedings of the international conference on high performance computing, networking, storage and analysis, SC '13 (2013), 25:1–25:12.

[36]K. Woolfe, *Matrix product operator simulations of quantum algorithms*, Aug. 2015.

[37]E. D. Napoli, D. Fabregat-Traver, G. Quintana-Ortí, and P. Bientinesi, "Towards an efficient use of the blas library for multilinear tensor contractions", Applied Mathematics and Computation **235**, 454–468 (2014).

[38]Y. Hirata, M. Nakanishi, S. Yamashita, and Y. Nakashima, "An efficient conversion of quantum circuits to a linear nearest neighbor architecture", Quantum Info. Comput. **11**, 142–166 (2011).

[39]M. A. Nielsen and I. L. Chuang, *Quantum computation and quantum information: 10th anniversary edition*, 10th (Cambridge University Press, New York, NY, USA, 2011).

[40]J. D. Hunter, "Matplotlib: a 2d graphics environment", Computing in Science Engineering **9**, 90–95 (2007).

[41]*Itensor*, `http://itensor.org` (visited on 09/30/2017).

[42]S. Al-Assam, S. R. Clark, and D. Jaksch, "The Tensor Network Theory Library", ArXiv e-prints (2016).

[43]T. Dierks and E. Rescorla, *The transport layer security (tls) protocol version 1.2*, RFC 5246, `http://www.rfc-editor.org/rfc/rfc5246.txt` (RFC Editor, Aug. 2008).

[44]A. G. Fowler, S. J. Devitt, and L. C. L. Hollenberg, "Implementation of shor's algorithm on a linear nearest neighbour qubit array", Quantum Info. Comput. **4**, 237–251 (2004).

[45]M. Frigo, "A fast fourier transform compiler", SIGPLAN Not. **39**, 642–655 (2004).

[46]R. L. Graham, T. S. Woodall, and J. M. Squyres, "Open mpi: a flexible high performance mpi", in *Parallel processing and applied mathematics: 6th international conference, ppam 2005, poznań, poland, september 11-14, 2005, revised selected papers*, edited by R. Wyrzykowski, J. Dongarra, N. Meyer, and J. Waśniewski (Springer Berlin Heidelberg, Berlin, Heidelberg, 2006), pp. 228–239.

[47]S. Beauregard, "Circuit for Shor's algorithm using 2n+3 qubits", eprint arXiv:quant-ph/0205095 (2002).

[48]C. Zalka, "Fast versions of Shor's quantum factoring algorithm", eprint arXiv:quant-ph/9806084 (1998).

[49]P. Gossett, "Quantum Carry-Save Arithmetic", eprint arXiv:quant-ph/9808061 (1998).

[50]J. Proos and C. Zalka, "Shor's discrete logarithm quantum algorithm for elliptic curves", Quantum Info. Comput. **3**, 317–344 (2003).

[51]S. Aaronson, "P $\stackrel{?}{=}$ NP", in *Open problems in mathematics*, edited by J. F. Nash Jr. and M. T. Rassias (Springer International Publishing, Cham, 2016), pp. 1–122.

[52]S. Aaronson and A. Arkhipov, "The computational complexity of linear optics", in Proceedings of the forty-third annual acm symposium on theory of computing, STOC '11 (2011), pp. 333–342.