Simulating Noisy Diamond Quantum Computers

Angus Mingare

A thesis submitted for the degree of
Bachelor of Science with Honours in Physics at
The Australian National University

October 2021

Declaration

This thesis is an account of research undertaken between February 2021 and October 2021 at the Research School of Physics, The Australian National University, Canberra, Australia.

Except where acknowledged in the customary manner, the material presented in this thesis is, to the best of my knowledge, original and has not been submitted in whole or part for a degree at any other university.

Angus Mingare

October 2021

Acknowledgements

I would like to start by thanking my supervisor, Marcus. I have greatly valued your wealth of knowledge, commitment, and ability to keep me (relatively) on track. From overcoming results that make no sense, to working through a lockdown, there wouldn't be a thesis without you.

This gratitude extends to my fellow NV-honours fellows, Milan and Michael, and the whole diamond research group for providing a welcoming research environment and exposing me to some very cool physics. And thanks goes to YunHeng for your help with noise models and reading some very messy derivations.

Throughout this year, I have greatly enjoyed the opportunity to work with some of the excellent team at Quantum Brilliance. In particular, I want to thank Nariman and Simon for generously helping me out with my many software-related questions. Your expertise has been invaluable.

It has been a tough year, but it has been made easier by the unwavering support and encouragement of my friends and family. So this thanks is for all of you (even if you never read this). I'd also like to shout out Tim for some last-minute edits.

Finally, a special mention to mum and dad, I wouldn't be here without you.

Abstract

Quantum computers have enormous promise in areas as diverse as medicine, finance, defence and information security. NV-centres in diamond are a promising platform for quantum computing due to their long coherence times at room temperature. However, current quantum computers are characterised by their noisy operation and do not provide any quantum advantage. Classical simulations can be used to provide feedback on various sources of noise to guide future engineering designs. Simulations can also be used to benchmark real quantum computers, and develop new quantum algorithms. Unfortunately, current noisy simulators are phenomenological and limited in scale. Hence they often poorly represent the general behaviour of real quantum computers, limiting their utility.

In this thesis I work towards developing an accurate, efficient simulator of noisy diamond quantum computers, with results generalisable to other architectures. I first compare the performance of two state of the art simulation methods to identify which is the most efficient. By performing runtime comparisons, I demonstrate that a leading Schrodinger simulator outperforms a leading tensor network simulator when performing high fidelity simulations, at least for fewer than 30 qubits. Next, I derive and validate a noise model based on decoherence and single-qubit control errors in diamond quantum computers. By simulating single-qubit circuits, I show that a depolarising channel produces simulations with a higher fidelity than the derived model. I attribute this to the small errors in single-qubit circuits favouring the fitted depolarising channel, which would not hold for multi-qubit circuits. I conclude that while the derived model requires further developments, it still offers immediate advantages over phenomenological models.

Contents

Chapter 1

Introduction

The development of quantum mechanics in the early twentieth century led to a technological revolution. Today, everything from electronics, to lasers and telecommunications, to MRI, is underpinned by quantum properties. As our ability to control quantum systems continues to improve, we are poised to develop a new wave of quantum technologies [1]. Among these, quantum computers have the greatest potential to revolutionise society is quantum computers. Quantum computers are devices that leverage the entanglement and superposition of quantum systems (called qubits) to gain a *quantum advantage*. That is, quantum computers offer up to exponential speed ups over classical computers when performing particular tasks.

The idea of using quantum systems to perform computations has been around since at least the 1970s with the development of quantum information theory [2] [3]. Then, in 1980, Beinoff described the first model of a quantum computer [4]. Despite this early work, quantum computing research didn't gain much traction until 1982, when Richard Feynman described the possible benefits of using quantum systems to simulate quantum systems [5]. The following decades saw interest in quantum computers continue to grow. Notably, Shor's algorithm for integer factorisation [6], and Grover's searching algorithm [7] proved that, in theory, quantum computers are able to outperform any known classical algorithms for particular, important tasks. Today, potential applications continue to emerge and suggest that quantum computers will be useful in a diverse range of areas

such as solving optimisation problems in finance and defence [8] [9], and modelling quantum systems for faster drug design and improved industrial chemistry processes [10] [11]. In short, the promise of quantum computers is enormous, and research conducted by major companies, government bodies, and numerous start-ups continues to bring quantum advantage closer to reality.

1.1 Noisy Intermediate-Scale Quantum Devices

Several designs have been proposed and tested for quantum computers, including using superconducting qubits [12], trapped ions [13], photons [14], and Nitrogen-Vacancy (NV) centres in diamond [15]. Each architecture has its own advantages and disadvantages. In the case of diamond quantum computers, a unique advantage is that NV-centre qubits have long coherence times at room temperature. Therefore diamond quantum computers can be operated at room temperature, compared to the cryogenic temperatures required by some other architectures. This simplifies the operation of the quantum computer, makes it more energy efficient, and allows it to be compact. Diamond quantum processors have already been used to demonstrate quantum error correcting codes [16], quantum algorithms [17] and simulations of quantum systems [18]. On the other hand, the creation of useful NV-centre qubits is difficult. Fabrication techniques for diamond quantum computers is in very early development, and scalability has not been fully shown [15].

As it stands, the development of quantum algorithms is years ahead of the development of physical quantum computers. For instance, useful implementations of Shor's algorithm would require a quantum computer with thousands of qubits, whilst current quantum computers comprise tens of qubits. Shor's algorithm is a relatively extreme example; it is predicted that quantum computers with between 50 and 100 qubits will be able to demonstrate a useful quantum advantage [19]. Such a demonstration, in which a quantum computer performs a task

faster than any known classical algorithm, is called *quantum supremacy*. Quantum supremacy is an important and highly sought after threshold in quantum computing research.

Quantum supremacy was claimed in 2018 by a team of researchers at Google [20]. Using their 53-qubit superconducting quantum computer, they performed a computation in 200 seconds that they say would take over 10,000 years on the world's most powerful classical computer. However, the task they performed (simulating random quantum circuits) has no real-world application, so achieving a *useful* quantum advantage remains a near-term goal [21].

The reason that quantum hardware is lagging behind quantum algorithms and applications is because building quantum computers is *hard*. Creating the quantum systems for a quantum computer requires incredibly precise and reliable nano-fabrication techniques. Furthermore, operation of a quantum computer requires precise control of these quantum systems. The techniques and tools required for fabrication and control of quantum computers have only existed for the past few decades and continue to require improvements today. Initially it was only possible to demonstrate few-qubit computers [22], and only very recently for intermediate-scale quantum computers with 50-100 qubits. Current quantum devices are characterised by their noisy operation and are hence termed Noisy Intermediate-Scale Quantum devices. This noise decreases the fidelity of computations, reducing the overall usefulness of the devices. There are two ways to deal with noise: allow noise to persist and correct any errors using error-correcting codes, or minimise the noise itself. Currently the cost of error-correcting codes is high (requiring many extra qubits), so near-term efforts are focused on decreasing noise at all stages of operation, from initialisation to readout.

1.2 The Case for Simulation

Noisy simulations are a powerful tool in the development of quantum computers. Simulations can be used to identify principal noise sources and their implication for the performance of the device. Thus they may be used to guide future engineering choices to reduce noise, and to optimise the compilation of quantum algorithms to mitigate the impact of errors. Simulations can also be used to validate and benchmark the performance of quantum computers, both against other quantum computers and against classical computers. This allows us to identify or predict the threshold of quantum advantage in different real-world applications. Most current quantum computer simulators either do not incorporate noise, poorly represent the behaviour of physical quantum computers, or are limited in scale [23] [24]. This is because the noise models used by these simulators are phenomenological, and the methods used to implement them are generally inefficient. Phenomenological noise models have no connection to the physical sources of noise affecting a device's performance. Due to this, these noisy simulators often do a good job at describing particular circuits run on particular hardware (the circuits that are used as training data for the model), but do not generalise well to other circuits or other devices. Furthermore, since the errors produced by phenomenological models are not related to the underlying noise sources, they provide no information about how to improve the quantum computer hardware to minimise the effects of noise.

1.3 Outline of this Thesis

This thesis addresses two problems in the development of noisy simulations of diamond quantum computers. Firstly, to compare current state of the art simulators to identify the most efficient. Secondly, to develop a noise model for diamond quantum computers that incorporates the underlying noise, to be implemented

in existing simulators. Achieving these aims would provide the foundations for an accurate and efficient noisy simulator for diamond quantum computers. The rest of the thesis is structured as follows.

In chapter 2, I give a brief review of the background theory necessary for the rest of the thesis.

I achieve the first aim in chapter 3 by using existing state of the art simulators to simulate noiseless quantum computers. I am interested in simulators that can describe quantum computers that can perform any quantum algorithm, *universal* quantum computers. There are two main simulation methods: the brute-force Schrodinger simulation method, and the more abstract class of tensor network methods. Tensor network methods have gained popularity in the last few decades as they can be used to efficiently simulate some quantum systems. However, the efficiency of tensor networks as a completely general quantum computing simulator is not present in the literature. Furthermore, comparisons of the efficiency between tensor networks and Schrodinger simulations rely mostly on complexity-theoretic arguments, rather than demonstrations using useful quantum circuits that we actually want to simulate. I address both of these issues in this chapter and determine the most efficient simulation method for general purpose quantum computing research.

I achieve my second aim in chapters 4 and 5 by deriving and validating a novel model describing the noise in diamond quantum computers. First, in chapter 4, I demonstrate the derivation methodology for decoherence and single-qubit gate errors, but it can be generalised to other sources of noise, or other quantum computing architectures. Then, in chapter 5, I validate the derived noise model by comparing it to a standard phenomenological model, and demonstrating how it may be used to provide feedback on sources of noise in the operation of the quantum computer.

I summarise the conclusions of this thesis and describe possible avenues for future work in chapter 6.

Chapter 2

Background Theory

In this chapter I give a brief review of the background theory necessary to the rest of this thesis. In Section 2.1 I start with the basic ideas behind quantum computing, including notation conventions that I use throughout this thesis. I provide an outline of the NV-centre in diamond and its use as a quantum computing architecture in section 2.2. Then, in section 2.3, I describe some of the common representations of quantum noise and standard techniques of converting between these representations. Sections 2.2 and 2.3 therefore provide the necessary theory and techniques that I use in chapter 4 to derive a noise model for diamond quantum computers. Finally, in section 2.4 I discuss some of the state of the art methods used to classically simulate quantum computers, both with noise and without noise, that I use in my investigations in chapters 3 and 5.

2.1 Quantum Computing Fundamentals

The information in this section is drawn from the foundational textbook by Nielsen and Chuang [25].

2.1.1 Qubits

The fundamental unit of a classical computer is the *bit*; a single piece of binary information carried as a 0 or 1. In a quantum computer, the fundamental unit is referred to as a *qubit* - a quantum bit. Whereas a classical bit can only exist as

a 0 or 1, a qubit, denoted by $|\psi\rangle \in \mathbb{C}^2$, can exist as an arbitrary superposition of the *computational basis states* $|0\rangle$ and $|1\rangle$,

$$|\psi\rangle = a\,|0\rangle + b\,|1\rangle\,, \tag{2.1}$$

such that $|a|^2 + |b|^2 = 1$. The states $|0\rangle$ and $|1\rangle$ can be represented by the usual basis states of $\mathbb{C}^2$,

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \qquad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}. \tag{2.2}$$

An alternative, often useful, description of a qubit is called the *Bloch sphere* representation. By parametrising $|\psi\rangle$ as

$$|\psi\rangle = \cos\frac{\theta}{2}\,|0\rangle + e^{i\phi}\sin\frac{\theta}{2}\,|1\rangle\,, \tag{2.3}$$

we can think of the state of the qubit as a point on the Bloch sphere as shown in figure 2.1. Applying a unitary operation to a qubit is equivalent to a rotation of the Bloch sphere. The state of n qubits is described by the tensor product of

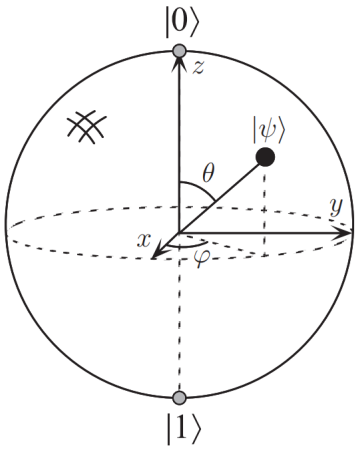


Figure 2.1: The state of a qubit, $|\psi\rangle$, can be described by a point on the Bloch sphere. From [25]

the individual qubit states,

$$|\Psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle \otimes ... \otimes |\psi_n\rangle \,. \tag{2.4}$$

The n-qubit state is therefore represented by a length 2^n vector.

2.1.2 Quantum Circuits

Quantum computers utilise the controlled evolution of a collection of qubits to solve problems. The method used to solve a particular problem is called a quantum algorithm. A general quantum algorithm consists of three stages:

1. **Initialisation:** First, the collection of qubits are prepared in some known initial state, generally the ground state $|00, ...0\rangle$.

2. **Evolution:** Next, a series of unitary operations are performed on the qubits and are designed to solve the problem being considered. In analogy to classical computing, these operations are called *gates* and the set of gates applied to the qubits is called a *circuit*. The number of layers of gates in a circuit is referred to as *circuit depth*.

3. **Readout:** Finally, in a well-designed quantum algorithm, the solution to the problem can be found by measuring the final state of the qubits.

Any computation that can be performed by a classical computer can be performed by a quantum computer. The converse is also true in theory, given enough time. The advantage of using a quantum computer is that by leveraging the superposition and entanglement of qubits, quantum computers can perform some tasks up to exponentially faster than classical computers.
Quantum circuits are often depicted graphically as circuit diagrams. The standard notation for these diagrams is shown in table 2.1. Some commonly used gates and their graphical representations are given in table 2.2. Just as n-qubit

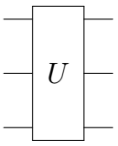
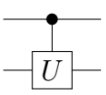
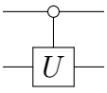
Circuit Component	Graphical Notation	Description
Wire		Qubits travel along wires from left to right.
Single-qubit gate		Gates acting on a qubit are represented by boxes drawn on that qubit's wire
Multi-qubit gates		This notation extends naturally to gates acting on multiple qubits
Conditional gate I		The filled-in circle indicates a control qubit. This means that U is applied to the target qubit if and only if the control qubit is in the $\|1\rangle$ state.
Conditional gate II		The empty circle also indicates a control qubit. However, now the operation U is applied to the target qubit if and only if the control qubit is in the $\|0\rangle$ state.

Table 2.1: Graphical notation for quantum circuit diagrams. U is some unitary operation.

states are represented by length 2^n vectors, n-qubit gates are represented by $2^n \times 2^n$ matrices. As an example, figure 2.2 shows the circuit that performs the quantum Fourier transform (QFT) - the quantum analogue of the discrete Fourier transform. The QFT is an important part of quantum algorithms including Shor's algorithm for integer factorisation.

A *universal quantum computer* is a quantum computer that is able to perform

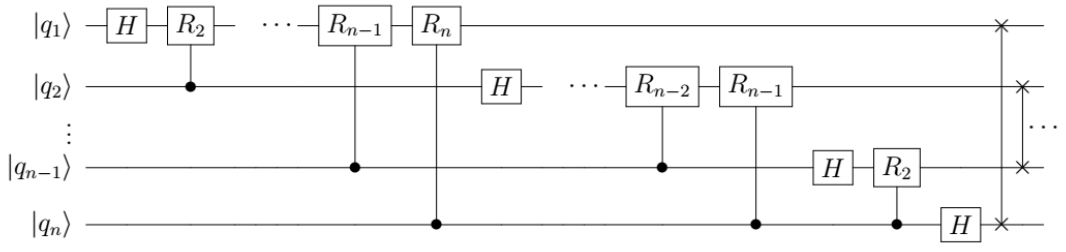


Figure 2.2: A quantum circuit diagram depicting the quantum Fourier transform (QFT).

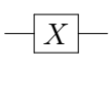
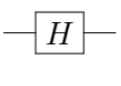
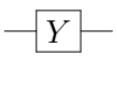
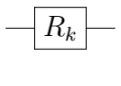
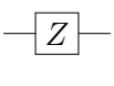
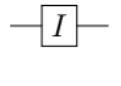
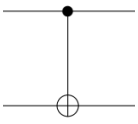
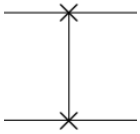
Gate	Graphical Notation	Matrix Form	Gate	Graphical Notation	Matrix Form
X		$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$	H		$\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$
Y		$\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$	R_k		$\begin{pmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^k} \end{pmatrix}$
Z		$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$	I		$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$
CNOT		$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$	SWAP		$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

Table 2.2: Some commonly used gates and their graphical representations.

any quantum computation. Any quantum computation on any number of qubits can be generated from a finite collection of gates, called a *universal set*. This seems counter-intuitive at first since there are an uncountably infinite set of unitary matrices that could act on the qubits. However, the idea is that a finite sequence of gates from the universal set can approximate any unitary operation to arbitrary accuracy. The Solovay-Kitaev theorem [26] ensures this can be done efficiently. Therefore, we only need to be able to physically implement a finite set of gates to build a universal quantum computer.

2.2 The Nitrogen-Vacancy Centre in Diamond

One of the proposed designs for physical quantum computers is based on the NV-centre in diamond [15]. A conceptual design for a diamond quantum computer is shown in figure 2.3. The NV-centre is a defect in the diamond lattice where a pair of neighbouring carbon atoms are replaced by a nitrogen atom and a vacancy. The NV-centre introduces a nuclear and an electronic spin degree of freedom into the diamond lattice. The nuclear spin of the nitrogen atom is a two-level system

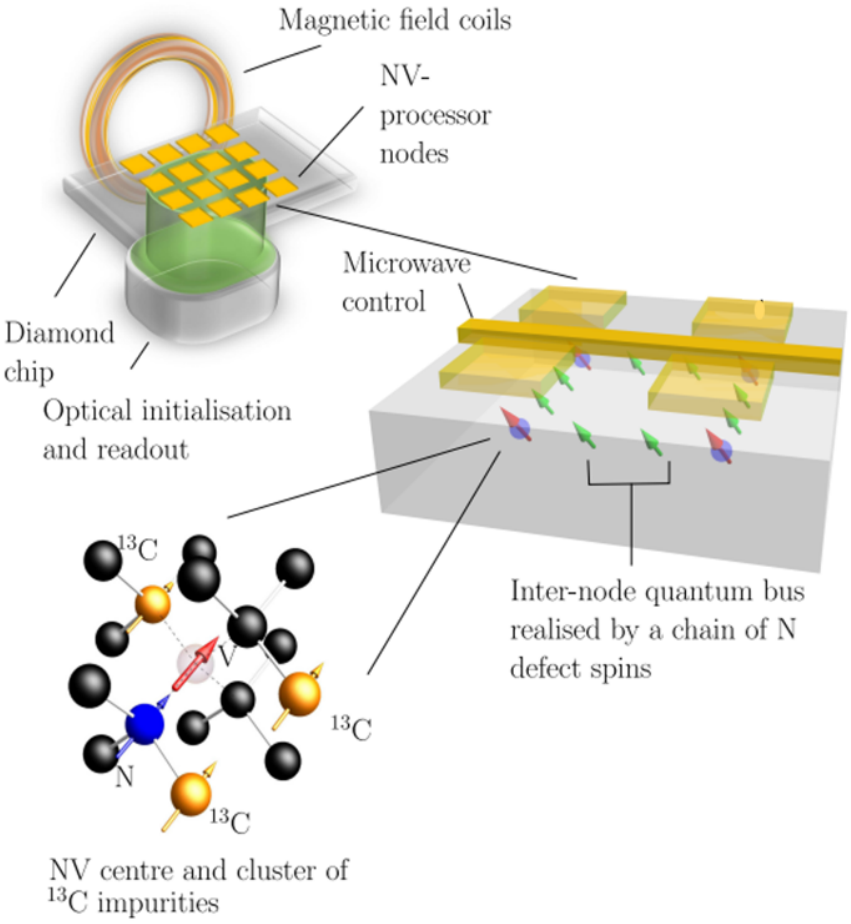


Figure 2.3: A conceptual design of a diamond quantum computer. Qubit clusters comprise an NV-centre and implanted magnetically coupled carbon-13 nuclei (bottom). These clusters are arranged on a diamond chip and are coupled through (e.g.) a spin chain of substitutional nitrogen nuclei. Gates are effected by surface microwave control systems (middle). Initialisation and readout are achieved through an optical system placed below the diamond chip (top). From [27]

and can therefore be used as a qubit in a quantum computer. In addition to the nitrogen nucleus, the density of carbon-13 nuclei in the diamond lattice will generally be artificially increased near the NV-centre to give a cluster of hyperfine-coupled nuclear spins that can all be used as physical qubits. This coupling is achieved through an externally applied magnetic field. There have been several proposed methods to couple individual clusters of qubits to neighbouring clusters, such as using a nuclear spin chain of substitutional nitrogen nuclei.

For universal quantum computing, we need to be able to initialise the qubit register, perform single-qubit and multi-qubit gates, and readout the final state. Optical initialisation and readout of the nuclear qubits are achieved using the uncoupled electron of the NV-centre as a quantum bus (a system that transports quantum information through entanglement). Further details of initialisation and readout can be found in [28] but are omitted here as they are not relevant to the rest of the thesis. To perform single-qubit gates, radio-frequency (RF) pulses can be used to drive arbitrary rotations of the Bloch sphere and hence any single-qubit gate. Microwave pulses can be used to implement a controlled-Z gate between any two qubits. Therefore, in practice, it is relatively straightforward to implement gates from the set

$$\{R_x(\theta), R_y(\theta), CZ\}, \tag{2.5}$$

where $R_x(\theta)$ and $R_y(\theta)$ are rotations of the Bloch sphere by angle θ about the x and y axes respectively, and CZ is the controlled-Z gate operation. Table 2.3 gives explicit descriptions as well as their graphical representation. It can be shown that this set of gates is universal [25].

Diamond is an exciting architecture for quantum computers. NV-centre qubits possess long coherence times at room temperature, removing the need for large and expensive cooling systems that are required for other proposed architectures. Furthermore, the control systems described are above are relatively simple to

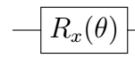
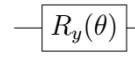
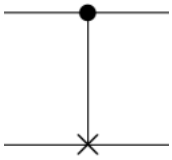
Gate	Graphical Notation	Matrix Form
$R_x(\theta)$		$\begin{pmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix}$
$R_y(\theta)$		$\begin{pmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix}$
CZ		$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}$

Table 2.3: The universal set of gates used in diamond quantum computers.

implement. However, fabrication of quantum processing units based on NV-centres in diamond is still in the early stages of development and many problems remain. For instance, NV-centres must be artificially created with a high density for scalable quantum computing, but with a regular spacing so that RF and microwave pulses targeted at a particular NV-centres do not affect other NV-centres. Other current areas of research include finding the most reliable way to couple qubit clusters, and how to maximise gate fidelities. There are still numerous challenges to overcome before diamond is a scalable, fault-tolerant hardware, however all the required components have been demonstrated [15].

2.3 Describing Noise

Current quantum computers are limited by noise. Since we need to be able to interact with the quantum computer during initialisation and readout, we cannot totally isolate the quantum computer from the environment. Therefore, quantum computers are inevitably open quantum systems and so are best studied through the density matrix formalism. The theory of open quantum systems is well-developed elsewhere (such as within the references given hereafter). I will only

provide (without proof) the tools used within this thesis.

2.3.1 Kraus Representation

To start, we will follow [29] and first consider a density matrix ρ. A noise process is given by some superoperator ξ such that

$$\rho_{in} \mapsto \rho_{out} = \xi(\rho). \tag{2.6}$$

Now, both ρ_{in} and ρ_{out} must be valid density matrices. That is, they must be hermitian, unit trace, positive-definite operators. This imposes some restrictions on ξ. Indeed, to be a valid quantum operator, ξ must satisfy the following properties,

1. Complete Positivity: A positive operator is one that maps positive elements to positive elements. For ξ to be completely positive we must have $(\xi \otimes I_k)$ is a positive operator for all k,

2. Trace-Preserving: That is $Tr(\xi(\rho)) = Tr(\rho)$ for all ρ.

We then have the following theorem:

Theorem 1 (Kraus Representation)**.** *Consider a d-dimensional quantum system represented by the density matrix ρ. Any Completely-Positive, Trace-Preserving (CPTP) map $\rho \mapsto \xi(\rho)$ can be written in the form*

$$\xi(\rho) = \sum_{i=1}^{N} K_i \rho K_i^\dagger,$$

for some $N < \infty$, where

$$\sum_{i=1}^{N} K_i^\dagger K_i = I.$$

The operators K_i are called Kraus operators.

2.3.2 Master Equation in Lindblad Form

For closed quantum systems, the density matrix evolves according to the von-Neumann equation,

$$\dot{\rho} = -i[H, \rho], \tag{2.7}$$

where H is the system Hamiltonian, $[\cdot]$ denotes the commutator and we are in units such that $\hbar = 1$. Equivalently, by defining the evolution operator

$$U = e^{-iHt}, \tag{2.8}$$

we have

$$\rho(t) = U\rho U^{\dagger}. \tag{2.9}$$

Now, consider the Kraus representation for the time evolution of an open quantum system,

$$\rho(t) = \sum_{i=1}^{N} K_i \rho(0) K_i^{\dagger}. \tag{2.10}$$

By considering $\rho(t + dt)$, it is possible to derive a generalisation of the von-Neumann equation for open quantum systems,

$$\dot{\rho} = -i[H, \rho] + \mathcal{L}_D(\rho), \tag{2.11}$$

where $\mathcal{L}_D$ is a superoperator that describes the decoherent part of the density matrix evolution. It can be written as

$$\mathcal{L}_D(\cdot) = \sum_{i=1}^{N} \gamma_i \left(L_i L_i^{\dagger} - \frac{1}{2}\{L_i^{\dagger} L_i, \cdot\} \right), \tag{2.12}$$

where $\gamma_i > 0$ and the *Lindblad operators* L_i are related to the Kraus operators in the small time limit by

$$K_1 = I + \left(-iH - \frac{1}{2} \sum_{i=2}^{N} L_i^{\dagger} L_i \right) dt, \tag{2.13}$$

$$K_i = \sqrt{\gamma_i} L_i \sqrt{dt}, \qquad i \geq 2. \tag{2.14}$$

Equation (2.11) is called a master equation in Lindblad form and provides another convenient description of open quantum systems. It is important to note that in the derivation of the master equation in Lindblad form, it is assumed that the quantum system is Markovian. This is not always true for quantum computers since we can have information fluctuating between the system and the environment, but in general it is a very good approximation. More precisely, it is a good approximation when the time evolution we care about is much longer than the time scale of the system-environment interactions. Another assumption of the Lindblad form of the master equation is that the system and environment are weakly interacting. The validity of this assumption for our purposes will be discussed in chapter 4.

2.3.3 Choi Matrix

There is one more representation of quantum operations that we will make use of: the Choi matrix. First, following [30], we cast the Lindblad form of the master equation as a matrix differential equation in terms of the vectorised density matrix $\tilde{\rho}$,

$$\rho = \begin{pmatrix} \rho_1 & \rho_2 & \cdots & \rho_d \end{pmatrix} \mapsto \tilde{\rho} = \begin{pmatrix} \rho_1 \\ \rho_2 \\ \vdots \\ \rho_d \end{pmatrix}, \tag{2.15}$$

where ρ_i denotes the i-th column of ρ. The matrix differential equation can be written,

$$\dot{\tilde{\rho}} = (\mathcal{G} + \mathcal{H})\tilde{\rho}, \tag{2.16}$$

where

$$\mathcal{G} = \sum_{m=1}^{N} \bar{L}_m \otimes L_m - \frac{1}{2} I \otimes (L_m^\dagger L_m) - \frac{1}{2} (\bar{L}_m^\dagger \bar{L}_m) \otimes I, \tag{2.17}$$

$$\mathcal{H} = -i(H \otimes I - I \otimes H), \tag{2.18}$$

and the overbar denotes complex conjugation. Solving equation (2.16) yields

$$\tilde{\rho}(t) = e^{(\mathcal{G}+\mathcal{H})t} \tilde{\rho}(0) \equiv M \tilde{\rho}(0) \tag{2.19}$$

The matrix M is referred to as the evolution matrix of the system. It is related to the Choi matrix, χ, by a simple permutation of its elements [31],

$$M_{m+s(n-1),j+s(k-1)} = \chi_{(j-1)s+m,(k-1)s+n}, \qquad j,k,m,n = 1,2,...,d. \tag{2.20}$$

To understand why we care about the Choi matrix we need the following theorem [32]:

Theorem 2 (Choi-Jamiolkowski Isomorphism)**.** *Consider a CPTP map acting between spaces of density matrices (possibly of different dimensions),* $\Phi : A \to B$*. Let* E_{ij} *denote the matrix whose* ij*-th entry is* 1 *and is* 0 *elsewhere, and consider* $\chi_\Phi = (\Phi(E_{ij}))_{ij} \in A \otimes B$*. Then the map*

$$\Phi \mapsto \chi_\Phi$$

is an isomorphism.

The notation I've used in the statement of the theorem is intentionally suggestive. It is the case that χ_Φ is the Choi matrix for the quantum operation Φ. Hence the Choi-Jamiolkowski isomorphism gives us a way to complete the circle and convert from the Choi matrix back to a CPTP map, which we know we can express in the Kraus representation by theorem 2.1. In lieu of a proof of theorem 2.2, I will outline the simple method to convert between the Choi matrix and the Kraus

representation as given in [31]. Consider the Choi matrix, χ, and diagonalise it in the standard way,

$$\chi = U_\chi D_\chi U_\chi^\dagger, \tag{2.21}$$

where U_χ is the matrix of eigenvectors of χ and D_χ is the diagonal matrix of eigenvalues of χ. Then form the matrix

$$e = U_\chi D_\chi^{1/2}. \tag{2.22}$$

The columns of this matrix, e_i, are vectorised Kraus operators of the CPTP map for the system. Returning them to matrix form using the inverse of the map in equation (2.15) gives us a collection of Kraus operators E_i. Thus, we can write

$$\rho(t) = \sum_i E_i \rho(0) E_i^\dagger. \tag{2.23}$$

Reversing the process to convert from the Kraus representation to a Choi matrix is straightforward. Form the matrix e of vectorised Kraus operators so that

$$\chi = ee^\dagger. \tag{2.24}$$

As a final remark, we observe that for a d-dimensional system, the Choi matrix has dimensions $d^2 \times d^2$. Hence, it has at most d^2 unique eigenvectors and therefore we get the following corollary:

Corollary 1. *Any CPTP map acting on d-dimensional density matrices can be expressed in the Kraus representation using at most d^2 Kraus operators.*

This concludes our discussion on dealing with noisy quantum systems. Figure 2.4 summarises the three representations we considered and the conversions between them.

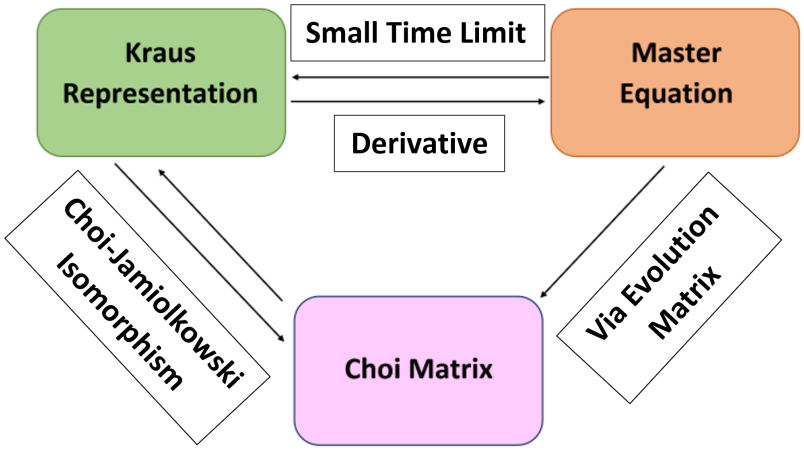


Figure 2.4: Three representations of noisy quantum systems and the relationships between them.

2.4 Classically Simulating Quantum Computers

There are many techniques used to classically simulate quantum computers. There is a broad class of quantum circuits, known as Clifford circuits, that can be efficiently simulated on a classical computer. This result is known as the Gottesman-Knill theorem [33]. However, it is clear that circuits from this class cannot display any quantum advantage, because if they did, it would be impossible to simulate them efficiently. In contrast, for most purposes, such as algorithm development and benchmarking quantum devices, we are only interested in simulating universal quantum computers. Hence, I don't consider Clifford simulators and instead focus on methods of simulating universal quantum computers. In chapter 3, I investigate two of the most popular methods for simulating noiseless quantum circuits: Schrodinger simulations and Matrix Product State (MPS) simulations [34]. Following a review of these simulation methods, I discuss methods of introducing noise into simulations.

2.4.1 Schrodinger Simulations

As we have already seen in section 2.1, a common representation for qubits is given by

$$|\psi\rangle = a\,|0\rangle + b\,|1\rangle = a\begin{pmatrix}1\\0\end{pmatrix} + b\begin{pmatrix}0\\1\end{pmatrix} = \begin{pmatrix}a\\b\end{pmatrix}. \tag{2.25}$$

The extension to n qubits is obtained by taking the tensor product of single-qubit states. Similarly, n-qubit gates are commonly written as $2^n \times 2^n$ matrices. A Schrodinger simulation of a quantum circuit is the brute-force method of evolving the qubits through the quantum circuit by matrix-vector multiplication.

Naive Schrodinger simulators therefore have memory requirements $\mathcal{O}(2^n)$ to store the state-vector, and operational costs $\mathcal{O}(2^{2n})$ to perform matrix-vector multiplication. Due to the exponential cost in n, Schrodinger simulations become infeasible for larger n. In particular, simulating more than around 40 qubits requires supercomputing resources and even current supercomputers cannot simulate far beyond $n = 50$.

Numerous techniques have been developed to improve Schrodinger simulators. These can be as simple as using sparse matrices in computations, to more sophisticated techniques such as circuit partitioning [35], and distributed memory [36]. The details are not important to this thesis, but exact Schrodinger simulations of up to $n = 64$ qubits have been performed.

2.4.2 Matrix Product State Simulations

MPS simulations fall into the broad class of tensor network simulation methods. In the last few decades, tensor network methods have gained popularity in simulating many-body quantum systems [37] [38]. In particular, they are a promising method for efficiently simulating quantum circuits with a large number of qubits, but relatively small circuit depth [39]. These methods work by converting a quantum circuit into a network of tensors, then contracting those

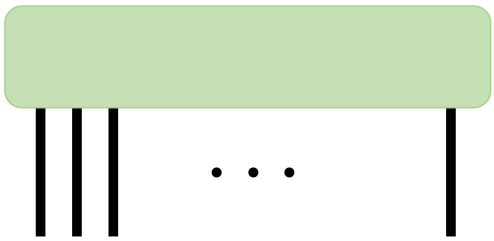


Figure 2.5: An order n tensor is represented by a box with n external legs.

tensors in specific ways to efficiently produce the output of the original quantum circuit. There exist many sub-varieties of tensor network simulations, for both state-vector and density matrix simulations, distinguished by either the way the tensors are constructed, or the way the tensors are contracted [40] [41]. However, they all share the same fundamental idea of replacing the large tensors required for Schrodinger simulations with a collection of smaller tensors. MPS is perhaps the most widely used tensor network method, certainly in the context of quantum computer simulations. It has several advantages over other tensor network methods, including having a simple construction and being able to calculate exact expectation values [42]. The following overview of MPS simulations draws together information from [42] and [43].

Tensor Networks

First, I will introduce a convenient graphical notation for tensor networks. Consider an order n tensor, $T_{\mu_1 \ldots \mu_n}$. Recall that this is a shorthand notation and represents the tensor

$$T = \sum_{\mu_1 \ldots \mu_n} T_{\mu_1 \ldots \mu_n} |\mu_1\rangle \ldots |\mu_n\rangle \,. \tag{2.26}$$

The number of values each index can take is referred to as the index dimension. An order n tensor is represented by a box with n external legs (one for each index), as shown in figure 2.5. The legs are often referred to as bonds. As an

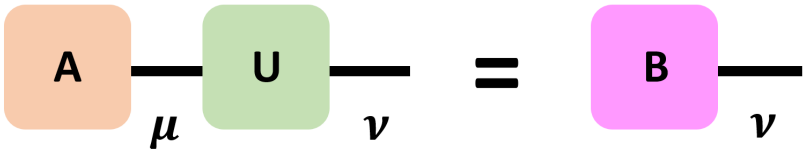


Figure 2.6: A tensor network representation of a matrix-vector multiplication. The shared bond represents the shared index and is summed over. The external bond represents the free index.

example, an n qubit system can be represented as a tensor $T_{\mu_1 \ldots \mu_n} \in \mathbb{C}^2 \otimes \ldots \otimes \mathbb{C}^2$ with each index $\mu_i \in \{0, 1\}$. This is an order n tensor with each index dimension equal to two. Hence it can be drawn as in figure 2.5. Similarly, single-qubit gates are generally written as 2×2 matrices, and so can be represented by an order two tensor with each index dimension equal to two, $U \mapsto U_{\mu_1 \mu_2}$. Hence it can be represented graphically as a box with two external legs. In tensor networks, n-qubit gates are thought of as order $2n$ tensors rather than matrices as we have previously described them.

Given two tensors, we can contract over a shared index by summing over it. For example, a matrix-vector multiplication can be written as

$$B_\nu = U_{\mu\nu} A_\mu \equiv \sum_\mu U_{\mu\nu} A_\mu, \tag{2.27}$$

using the usual Einstein summation convention. At this point it is worth noting that there are nuances with raised and lowered indices that extend to the graphical notation [43]. However, these aren't important to understanding the rest of this thesis so I won't mention them here, and will continue using the convention of writing all indices as subscripts with the implicit assumption that repeated indices are summed over.

In the graphical notation, a tensor contraction is depicted by a shared bond between two tensors. For example, figure 2.6 depicts the matrix-vector multiplication from equation (2.27).

Singular Value Decomposition

Now that we understand the graphical notation, we need one additional tool before we can discuss MPS explicitly. The Singular Value Decomposition (SVD) is a generalisation of the standard eigen-decomposition for non-square matrices. That is, given an $m \times n$ complex matrix M, there exists a decomposition of the form

$$M = U\Sigma V^\dagger, \tag{2.28}$$

such that U is an $m \times m$ complex unitary matrix, Σ is an $m \times n$ rectangular diagonal matrix with real, non-negative entries on the diagonal, and V is an $n \times n$ complex unitary matrix. The non-zero entries of Σ are called singular values. The number of singular values of Σ is referred to as its dimension, χ, and it is equal to the rank of M. Clearly

$$\chi \leq \min(m, n). \tag{2.29}$$

Furthermore, the SVD can always be done in such a way that the singular values appear in non-increasing order along the diagonal of Σ. Often, it is useful to truncate the SVD. This means only keeping the first k singular values. Consequently, you keep only the first k columns of U and the first k rows of $V^\dagger$. The justification for doing this is that it can be shown [44] that the resulting matrix is the best rank-k matrix approximation to M in terms of the Frobenius norm:

$$\|M\|_F = \sqrt{\sum_{ij} M_{ij}}. \tag{2.30}$$

Forming a MPS

Recall that the idea behind tensor networks is to rewrite large tensors such as the state-vector for the quantum system as a collection of smaller tensors. In the

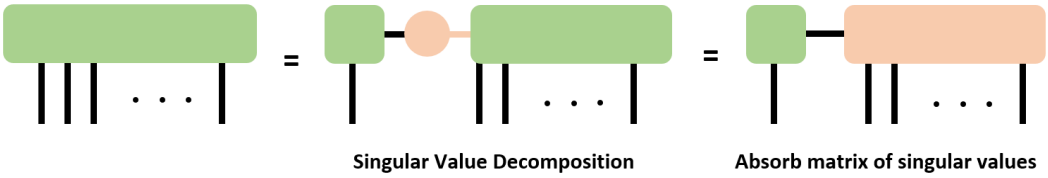


Figure 2.7: A SVD is performed on the $2 \times 2^{n-1}$ matrix representing the n qubit state, and then the matrix of singular values is absorbed, resulting in a 2×2 matrix and a $2 \times 2^{n-1}$ matrix.

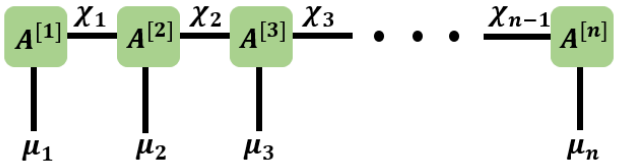


Figure 2.8: A series of SVDs is used to convert the state-vector of an n qubit system into a network of tensors known as a MPS. $\mu_1, ...\mu_n$ are the 2-dimensional external bonds representing the physical qubits and $\chi_1, ...\chi_{n-1}$ are the internal bonds.

case of MPS, we achieve this using the SVD. First, consider an n qubit state

$$|\Psi\rangle = \sum_{\mu_1 ... \mu_n} T_{\mu_1 ... \mu_n} |\mu_1 ... \mu_n\rangle \,, \tag{2.31}$$

drawn graphically in figure 2.5. The length 2^n state-vector for this state can be rewritten as a $2 \times 2^{n-1}$ matrix in a standard way (referred to as folding, or matricisation). We can then perform the SVD on this matrix and "absorb" the Σ matrix as shown in figure 2.7. Now we repeat a similar process on the remaining $n-1$ qubit state. A full mathematical description for the process can be found in [45]. The end result is given in graphical form in figure 2.8. Then, using the notation of figure 2.8, we can write

$$T_{\mu_1 ... \mu_n} = \sum_{\chi_1 ... \chi_{n-1}} A^{[1]}_{\mu_1 \chi_1} A^{[2]}_{\mu_2 \chi_1 \chi_2} ... A^{[n]}_{\mu_n \chi_{n-1}}. \tag{2.32}$$

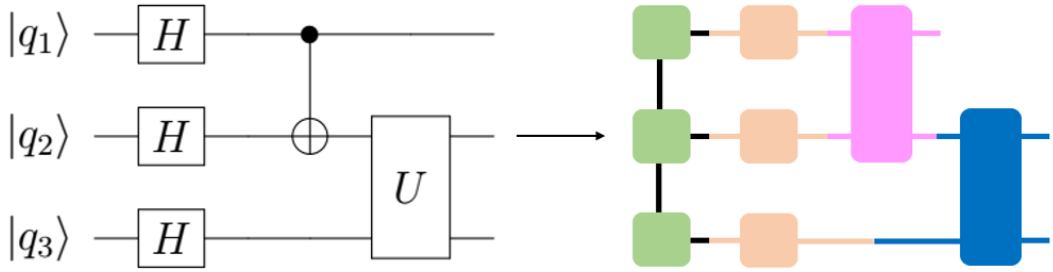


Figure 2.9: On the left is a randomly chosen quantum circuit diagram. On the right, the circuit is converted into MPS form. The green boxes correspond to the MPS representation of the input state, the orange boxes correspond to the first layer of H gates, the pink box corresponds to the CNOT gate, and the blue box corresponds to the two-qubit U gate.

The maximum dimension over all internal bonds is called the bond dimension of the MPS and is denoted by χ. A consequence of equation (2.29) is that, for a system of n qubits, we have

$$\chi \leq 2^{\lfloor n/2 \rfloor}. \tag{2.33}$$

MPS Simulations

The first step in an MPS simulation of a quantum computer is to prepare the input state and all gates in MPS form. The graphical notation for this is very similar to the quantum circuit diagram notation seen in section 2.1 and is shown in figure 2.9. Once we have constructed the MPS representation of the quantum circuit, we can obtain the output state by contracting over the indices between the state-vector and the gates. That is, we contract over the horizontal bonds in figure 2.9, from left to right. The output state will be a MPS of the same order as the input state. If you were to contract all internal bonds in the output MPS, you would obtain the same state-vector output of the quantum circuit as given by a straightforward Schrodinger simulation. However, the reason for using a MPS simulation is that we don't need to store the full state-vector. So, instead contracting the internal bonds of the output state, we contract the output MPS with an MPS representation of each of the basis states. This is equivalent

to taking an inner product and gives the probability distribution of the output state. The details are unimportant, but it is important to note that this is the most computationally expensive part of an MPS simulation. It can be shown [43] that the operational cost of an MPS simulation of n qubits is $\mathcal{O}(n\chi^3)$, and the memory requirements are $\mathcal{O}(n\chi^2)$.

The theory outlined in this section was all aimed at reaching this conclusion: the bond dimension is an important parameter in MPS simulations. Given any n-qubit state, we can find a perfect MPS representation for it, given a large enough bond dimension. Equation (2.32) guarantees that in the worst case this is $\chi = 2^{\lfloor n/2 \rfloor}$, and there are specific cases where it is much lower. On the other hand, the computational costs associated with MPS simulations can be made linear in n by truncating the bond dimension χ to some fixed constant. However, this introduces errors into the simulation. Therefore, there is a trade off between computational efficiency and simulation fidelity.

2.4.3 Including Noise

To include noise in simulations of quantum computers we generally need to consider density matrices rather than state-vectors. The state of the system will be represented by a density matrix and gates are applied by,

$$\rho \mapsto U\rho U^\dagger. \tag{2.34}$$

Then, at every point in the circuit we wish to include noise, we can use any of the three noise models described in section 2.3. For example, we can use a Kraus channel,

$$\rho \mapsto \sum_i K_i \rho K_i^\dagger. \tag{2.35}$$

This direct evolution method is analogous to the Schrodinger simulation method outlined above, except now the memory requirements are $\mathcal{O}(2^{2n})$ and the oper-

ational costs are $\mathcal{O}(2^{3n+1})$. There also exist density matrix analogues of MPS simulations, called matrix product density operator (MPDO) simulations [46] however these are not as well studied, and suffer some disadvantages compared with MPS simulations.

Recall that by the Choi-Jamiolkowski isomorphism, we can use an evolution matrix rather than a Kraus channel,

$$\tilde{\rho} \mapsto U\tilde{\rho}, \tag{2.36}$$

where $\tilde{\rho}$ is the vectorised density matrix. However in this case, U is a $2^{2n} \times 2^{2n}$ matrix so the computational requirements for this method are greater than using the Kraus channel.

The final option for including noise is to use a master equation,

$$\frac{d\rho}{dt} = -i[H, \rho] + \mathcal{L}_D(\rho), \tag{2.37}$$

and integrate over a small time period. Whilst this method can be accurate, density matrix solvers are infeasible for more than a few qubits.

Kraus models are the most used method of including noise in the literature as they are the most computationally efficient. Furthermore, in certain cases, the Kraus model can be used to incorporate noise into state-vector simulations (such as Schrodinger simulations and MPS simulations). This will be discussed further in section 3.3.

Chapter 3

Simulating Quantum Computers

Being able to simulate quantum computers is a powerful tool for quantum computing research, and this thesis aims to contribute to the development of improved simulations of noisy diamond quantum computers. As outlined in the introduction, there are two main motivations behind simulating quantum computers. The first is to identify principal sources of noise and provide feedback on their implications for device performance to improve quantum hardware designs. And the second is to validate and benchmark the performance of real quantum computers. A useful simulator for quantum computers should have high fidelity, and be efficient.

In this chapter, I address the first of my two aims: to compare current state of the art simulation methods to identify which is most efficient. I use IBM's Aer state-vector simulator as a representative for Schrodinger simulators because it is widely used and well-developed. Consequently, it is reliable and fast. For similar reasons, I also use Oak Ridge National Laboratory's Exascale Tensor Network (ExaTN) MPS simulator. I execute all simulations using Quantum Brilliance's Quantum Emulator (QBQE); an IDE to simulate quantum circuits that supports both the Aer simulator and the ExaTN-MPS simulator [47]. All simulations are performed using cloud computing resources through Amazon Web Services (AWS).

MPS simulators have gained popularity because the computational resources scale linearly with qubit number, assuming the bond dimension is truncated to a

fixed constant. In section 3.1, I devise a worst-case scenario for MPS simulators to investigate the validity of truncating the bond dimension in general. This allows me to determine the bond dimension that is necessary for the MPS simulator to produce outputs with an acceptable fidelity, greater than some threshold value. In section 3.2, I simulate important quantum circuits using the Aer and ExaTN-MPS simulators to compare their runtime. For these simulations, I ensure the bond dimension of the MPS simulator is set high enough so that its fidelity is guaranteed to be above the acceptable value. Since these investigations are concerned with trade off between fidelity and efficiency in MPS simulations, in this chapter I will only consider ideal quantum computers. This ensures the fidelity of the MPS simulator is being affected only by bond dimension and not by noise.

The results of this chapter will determine which simulation method should be preferred for general purpose quantum computing research.

3.1 Bond Dimension vs. Fidelity in MPS Simulations

3.1.1 Defining the Worst-Case Scenario

We know that given high enough bond dimension, MPS simulations are guaranteed to be perfect. We denote this value by χ_p so that

$$\chi_p \equiv 2^{\lfloor n/2 \rfloor}, \tag{3.1}$$

where n is the number of qubits involved in the simulation. Note that in some cases, the bond dimension required for a perfect simulation is smaller than χ_p.

For example, the circuit that prepares the maximally entangled state

$$\frac{1}{\sqrt{2}}(|00...0\rangle + |11...1\rangle), \tag{3.2}$$

can be represented perfectly by an MPS simulation using bond dimension $\chi = 2$ for all n [48]. By "perfect", we mean that the simulation produces an output that is identical to the expected behaviour of the quantum circuit. This is measured using a metric called fidelity. In this chapter, I use the standard definition of fidelity. Given the expected output $|\psi_e\rangle$ and a simulated output $|\psi_s\rangle$, the fidelity $\mathcal{F}$ is defined to be

$$\mathcal{F} = |\langle\psi_s|\psi_e\rangle|^2. \tag{3.3}$$

Therefore, a fidelity of 1 is perfect, and a fidelity of 0 implies there is no overlap between the simulated output and the expected output. For simulations to be useful, they should represent reality as closely as possible. So, they should have fidelity very close to 1, say greater than 0.99999 (the exact threshold will turn out to be irrelevant). We know that for bond dimension χ_p, MPS simulations have fidelity 1. However, χ_p is exponential in n which is a problem since, as previously discussed, the bond dimension also controls the computational cost of the simulation. Truncating the bond dimension at a fixed constant solves this problem at the cost of introducing errors into the simulation. This is common in the literature, as it is generally assumed that the errors introduced by truncating the bond dimension are small enough to be acceptable.

To test this assumption, let us construct a worst-case example. The following methodology, including the developed notion of a worst-case example for MPS is novel. Such a test is an important concept missing from the literature. I will design a simulation where I know we require bond dimension $\chi = \chi_p$ to achieve a fidelity of 1. Then by truncating the bond dimension at values lower than this I can see how quickly the fidelity drops below 1. To justify truncating the bond dimension, the fidelity should stay very close to 1 until the bond dimension gets

very low.

First, I introduce the following convenient notation from [42]. Excluding the terminal tensors, every tensor in an MPS is of order 3, so we write it as $A_{\mu\chi_1\chi_2}$ where μ corresponds to the external bond, and χ_1, χ_2 correspond to the internal bonds. Since the external bond represents the physical qubit, it can take only two values, 0 or 1. Hence we can represent $A_{\mu\chi_1\chi_2}$ by a matrix with vector entries. For example, the matrix

$$\begin{pmatrix} |0\rangle & 0 \\ 0 & |1\rangle \end{pmatrix}, \tag{3.4}$$

corresponds to the tensor

$$A_{0\chi_1\chi_2} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, \qquad A_{1\chi_1\chi_2} = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}. \tag{3.5}$$

In this notation, matrix-matrix multiplication works in the usual way with the multiplication of numbers replaced by the tensor product of vectors. For example,

$$\begin{pmatrix} |0\rangle & 0 \\ 0 & |1\rangle \end{pmatrix} \begin{pmatrix} |1\rangle & 0 \\ 0 & |0\rangle \end{pmatrix} = \begin{pmatrix} |01\rangle & 0 \\ 0 & |10\rangle \end{pmatrix}. \tag{3.6}$$

Now, we know that in the general case, performing successive SVDs on an n-qubit state, $|\psi\rangle$, yields an MPS chain of the following form:

$$|\psi\rangle = \begin{pmatrix} \cdot & \cdot \end{pmatrix} \begin{pmatrix} \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{pmatrix} \begin{pmatrix} \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{pmatrix} \cdots \begin{pmatrix} \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{pmatrix} \begin{pmatrix} \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \end{pmatrix} \begin{pmatrix} \cdot \\ \cdot \end{pmatrix}, \tag{3.7}$$

where the entries are vectors as described above. In some cases, some of these rows and columns may be zeros, in which case the bond dimension can be truncated and no information is lost so the fidelity remains 1. In other cases, some of these rows and columns will contribute less to the final state, depending on the size of the singular values in the SVD. Hence, if the bond dimension is truncated, a small amount of information is lost so the fidelity drops below (but stays very close to) 1. However, in the worst-case scenario, all of these rows and columns are equally important. We want to investigate how the fidelity is affected in this worst-case situation.

3.1.2 Constructing the Circuit

Consider the following MPS,

$$|\psi\rangle = \begin{pmatrix} |0\rangle & |1\rangle \end{pmatrix} \begin{pmatrix} |0\rangle & |1\rangle & |0\rangle & |1\rangle \\ |1\rangle & |0\rangle & |1\rangle & |0\rangle \end{pmatrix} \cdots \begin{pmatrix} |0\rangle & |1\rangle \\ |1\rangle & |0\rangle \\ |0\rangle & |1\rangle \\ |1\rangle & |0\rangle \end{pmatrix} \begin{pmatrix} |0\rangle \\ |1\rangle \end{pmatrix}, \tag{3.8}$$

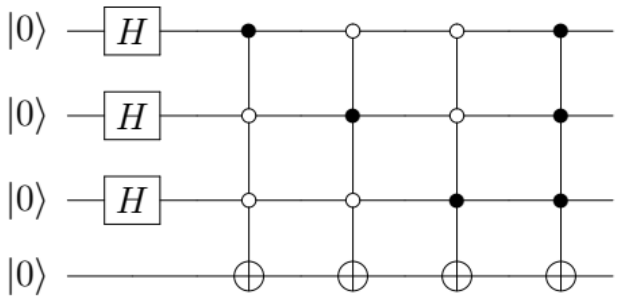


Figure 3.1: This circuit constructs an equal superposition of all 4-qubit states that contain an even number of 1s

where the matrix entries alternate between $|0\rangle$ and $|1\rangle$. For an even number of qubits, n, when this state is contracted it forms an unnormalised, equal superposition of the n-qubit states that contain an even number of 1s. For example, when $n = 4$ we have the superposition state

$$|\psi\rangle = |0000\rangle + |0011\rangle + |0101\rangle + |0110\rangle + |1001\rangle + |1010\rangle + |1100\rangle + |1111\rangle \,. \quad (3.9)$$

Notice that all rows and columns contribute equally to the final output. Hence, I predict that a circuit designed to produce this state (given the ground state as input) should be a worst-case scenario for an MPS simulator. That is, for bond dimension $\chi = \chi_p$ we achieve fidelity 1, but for any bond dimension $\chi < \chi_p$, the fidelity is less than 1. Indeed, as we shall see, simulation confirms this.

I will refer to the state described above as an even-1s state. A circuit to construct an even-1s state is relatively straightforward to build. For an n-qubit circuit, first apply a Hadamard gate to the first $n-1$ qubits. This produces an equal superposition of all $(n-1)$-qubit states and leaves the final qubit in the ground state $|0\rangle$. For those $(n-1)$-qubit states that already contain an even number of 1s, we don't do anything. For those $(n-1)$-qubit states that have an odd number of 1s, we need to flip the final qubit into the $|1\rangle$ state. We can achieve this with a sequence of controlled operations. An example for $n = 4$ is shown in figure 3.1. Physically we only have ability to perform single-qubit and two-qubit gates.

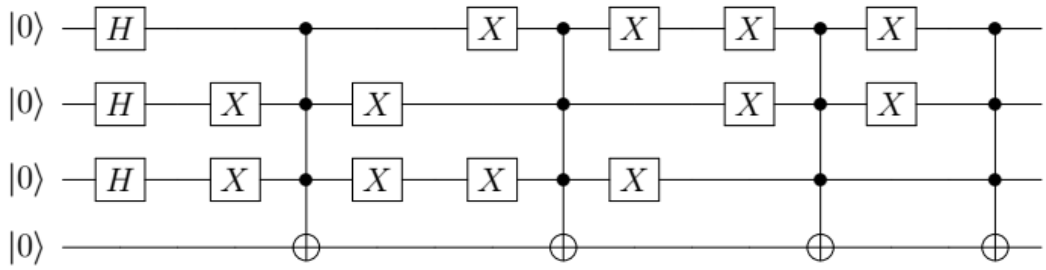


Figure 3.2: This circuit is equivalent to the circuit shown in figure 3.1.

Hence we need to decompose the above circuit into these gates. First, recall from table 2.1 that the solid circles indicate the operation is conditional on the qubit being in the $|1\rangle$ state, while the open circles indicate the operation is conditional on the qubit being in the $|0\rangle$ state. We can replace all open-controls by closed-controls, as long as we add an X gate either side of where the open-control used to be. The $n = 4$ case is shown in figure 3.2. Now we just need to know how to decompose $C^{n-1}NOT$ gates. That is, a NOT gate with $n-1$ controls. There are very simple ways to do this if we are allowed to introduce *ancillary* qubits, or simply *ancilla*. These are qubits that are used during the computation but do not contribute to the output state. They can be thought of as additional workspace for computations. Unfortunately, if we introduce ancilla we are no longer guaranteed that the circuit will be a worst-case example. Therefore, we need a way to decompose $C^{n-1}NOT$ gates without introducing ancilla, which is far more difficult. However, such a decomposition can be found in [49]. The method involves first performing a change of basis by applying the single-qubit gate,

$$M = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}, \tag{3.10}$$

to all qubits, and then applying the circuit shown in figure 3.3 before returning to the original basis by applying $M^\dagger$ to all qubits.

We now have a circuit that can produce the even-1s state from the ground state

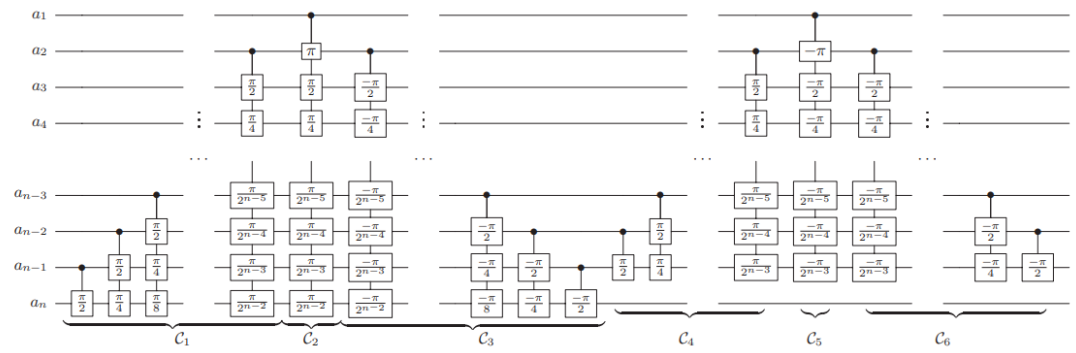
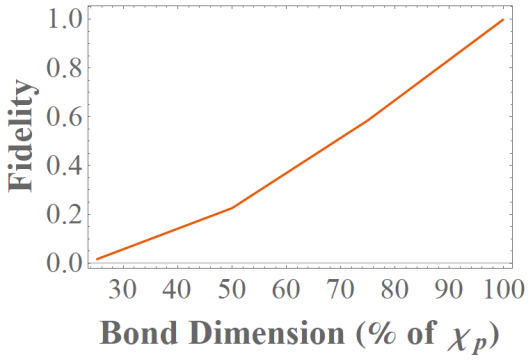


Figure 3.3: A circuit to perform a $C^{n-1}NOT$ gate using only single-qubit gates and single-control gates. All gates are controlled rotations about the x-axis with the angle of rotation given in the box. From [49].

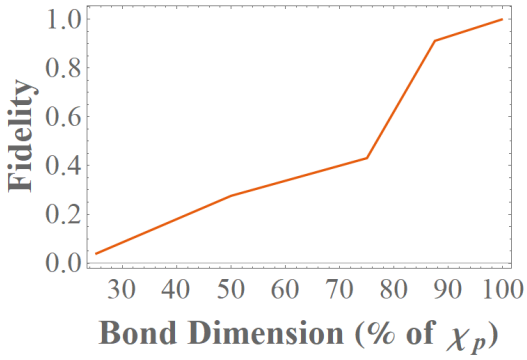
written in terms of single-qubit gates and single-control gates only. Different quantum computer architectures use different universal sets to implement arbitrary single-qubit and two-qubit gates. For example, to simulate a diamond quantum computer we would need to further decompose this circuit into gates from the set $\{R_x(\theta), R_y(\theta), CZ\}$, however there are standard ways to do this [25], so the details are omitted.

3.1.3 Running the Simulation

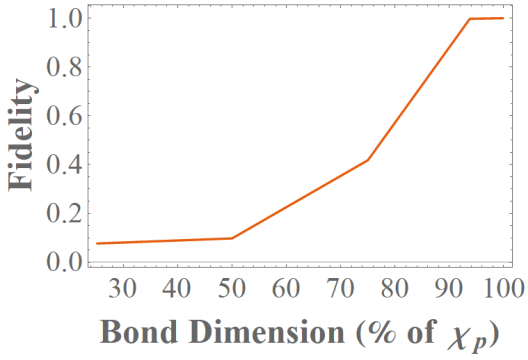
Even for $n = 4$, the even-1s circuit is very complicated. Indeed, the circuit depth grows very fast with qubit number n, making simulations very slow beyond just a few qubits. Using 16 virtual-CPUs on AWS with 64GB RAM, I was able to perform the simulation for 4, 6, 8, and 10 qubits in a reasonable amount of time. I performed the simulations taking 1,000,000 shots (samples of the output state) to obtain the probability distribution of the output. Using one million shots minimised the effects of shot noise. I then calculated the fidelity between this simulated distribution and the expected even-1s distribution using equation 3.3. By varying the bond dimension χ, I could investigate the relationship between bond dimension and fidelity for the even-1s circuit. In figure 3.4, I have plotted the fidelity against the bond dimension as a percentage of χ_p, $\frac{\chi}{\chi_p} \times 100\%$. The



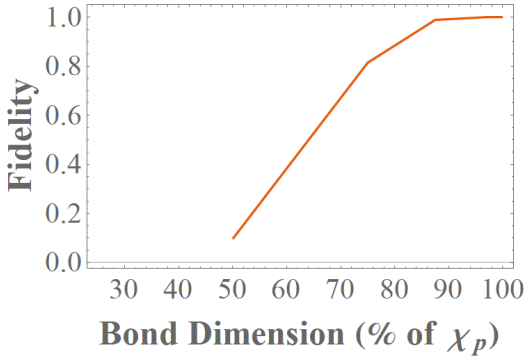
(a) $n = 4$



(b) $n = 6$



(c) $n = 8$



(d) $n = 10$

Figure 3.4: The fidelity of the MPS simulation for the even-1s circuit is plotted against the bond dimension as a percentage of χ_p.

first thing to notice is that the fidelity drops below 1 for all $\chi < \chi_p$, justifying the choice of this circuit as a worst-case example.

Furthermore, the fidelity drops below 1 *quickly* as the bond dimension is reduced below χ_p. However, as the number of qubits increases, we observe a flattening of the curve near χ_p. That is, for higher qubit numbers, the simulation fidelity stays very close to 1 for bond dimensions less than, but close to, χ_p. This is exactly the behaviour that would support the validity of truncating the bond dimension. For example, for $n = 10$ qubits we have $\chi_p = 32$, but truncating the bond dimension to $\chi = 31$ still yields a high fidelity simulation. We expect that this flattening of the curve continues for higher qubit number. Unfortunately, since χ_p is exponential in n, it does not appear as though this effect is large enough for a fixed truncation to be feasible. For instance, suppose we truncate the bond dimension at $\chi = 1000$, a reasonable value in practice. This is only 3% of χ_p for $n = 30$ and that percentage halves for every additional 2 qubits added. There is no evidence that the flattening of the curve will be able to match this exponential decrease at higher qubit number. However, future work should use supercomputing resources to test the even-1s circuit for higher qubit numbers to confirm this conclusion.

There are examples, such as the maximally-entangled state, where truncating the bond dimension yields simulations with sufficiently high fidelities for the circuits being considered. The even-1s example shows that this cannot be assumed to be true in general. The even-1s circuit was artificially constructed to be difficult for MPS simulators and has no known application. It might be the case that any useful circuit admits simulations with a fixed bond dimension. There are claims in the literature that truncating the bond dimension is appropriate for low-entanglement circuits [50]. However, as described above, the maximally entangled state can be perfectly represented with bond dimension $\chi = 2$ for any qubit number n, so the precise relationship between entanglement and required bond dimension is not clear in practice. Further research is required to elucidate which

properties of quantum circuits determines the bond dimension required for high fidelity MPS simulation.

The computational cost of MPS simulations scales linearly with qubit number, assuming the bond dimension is truncated to a fixed value. While this is valid for some circuits, I have shown that it is not valid for all circuits. Therefore, for general research purposes, including quantum algorithm design and testing, MPS simulators with truncated bond dimension should be used with caution. MPS simulators may still offer advantages if they are more efficient than Schrodinger simulations, and this will be investigated in the next section.

3.2 Aer vs. ExaTN-MPS

A quantum computer simulator should be high fidelity and efficient. In this section, I compare the efficiency of Schrodinger simulations with MPS simulations when performing high fidelity simulations, using runtime as the performance metric. Schrodinger simulations are always fidelity 1 whereas the fidelity of MPS simulations depends on the bond dimension. As shown in the previous section, to be guaranteed that the fidelity is sufficiently close to 1 for all n we cannot truncate the bond dimension to a fixed value. Therefore, for all MPS simulations in this section the bond dimension is chosen to be χ_p. This ensures the fidelity of the MPS simulations is 1. As mentioned in chapter 2, the memory requirements of Schrodinger simulations is $\mathcal{O}(2^n)$, compared to $\mathcal{O}(n\chi^2)$ for MPS simulations. In our case, $\chi = \chi_p$ so that the memory requirements for MPS simulations are $\mathcal{O}(n2^n)$, worse than Schrodinger simulations. However, the promise of MPS is in the required operational resources. For MPS this is $\mathcal{O}(n\chi^3) = \mathcal{O}(n2^{3n/2})$, which is better than the $\mathcal{O}(2^{2n})$ required for Schrodinger simulations. So, in theory the scaling behaviour indicates MPS simulations outperform Schrodinger simulations - even using the maximum bond dimension. However, MPS simulations come with significantly greater *overhead* required to initialise. In practice what this

means is that while MPS simulations may outperform Schrodinger simulations in the limit of large n, this will almost certainly not be the case for small n. Therefore, we predict the existence of a cross over point; a qubit number for which MPS simulations begin to run faster than Schrodinger simulations. This cross over may be different for different circuits. For MPS simulations to be useful, this cross over point should occur in the range of qubit numbers that we are able to simulate. I attempt to locate this cross over point for two practical quantum circuits.

3.2.1 GHZ State Preparation

Recall the maximally entangled state, is written as

$$|GHZ\rangle = \frac{1}{\sqrt{2}}(|00...0\rangle + |11...1\rangle), \tag{3.11}$$

for any number of qubits. This state is often referred to as the Greenberger-Horne-Zeilinger (GHZ) state. The GHZ state is an important state, used in several protocols in quantum communications and quantum cryptography [51]. The circuit to generate the GHZ state from the ground state is very simple and the $n = 4$ case is shown in figure 3.5. A single Hadamard gate is applied to the first qubit to put it in an equal superposition state. A cascading sequence of CNOT gates follows, ensuring that successive qubits remain in the $|0\rangle$ state conditional on the first qubit being in the $|0\rangle$ state, and flipping to the $|1\rangle$ state conditional on the first qubit being in the $|1\rangle$ state. I performed simulations of the GHZ circuit with both Aer and ExaTN-MPS for various number of qubits..Unfortunately, I encountered a bug in the ExaTN-MPS code preventing me from simulating 20 or more qubits. This issue was reported to Oak Ridge National Laboratory and has since been fixed, so future work should repeat these simulations to extend beyond 20 qubits for ExaTN-MPS. As it is, we see no cross over point. However, we know the theoretical functional form for the scaling behaviour of both

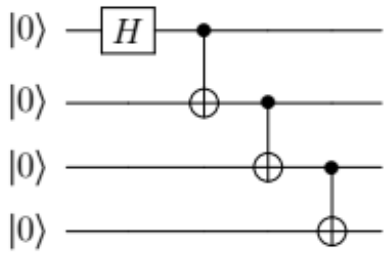


Figure 3.5: The circuit used to prepare the GHZ state from the ground state for $n = 4$ qubits.

Simulator	Expected Scaling	Fitting Function	a	b	c
Aer	$\mathcal{O}(2^{2n})$	$a + b2^{cn}$	0.029194	1.2771×10^{-7}	1.0722
ExaTN-MPS	$\mathcal{O}(n2^{3n/2})$	$a + bn2^{cn}$	0.95595	4.5374×10^{-5}	0.84129

Table 3.1: Fitting functions and parameters for the scaling behaviour of the Aer and ExaTN-MPS simulators for the GHZ circuit. These are plotted in figure 3.6.

Schrodinger simulations, $\mathcal{O}(2^{2n})$, and MPS simulations, $\mathcal{O}(n2^{3n/2})$. By fitting functions to the data I have, I can extrapolate to find an estimate cross over point for the GHZ circuit. The fitting functions and their parameters are given in table 3.1. The simulation runtime data and the fitting functions are plotted together in figure 3.6. The constant a in the fitting functions accounts for any initialisation overhead and as predicted this is larger for the MPS simulator. We also notice that the scaling behaviour is much better than the theoretical prediction in both cases. We do not currently understand this completely, however it is likely due to both simulators employing various techniques to improve upon the naive simulation methods they are based upon. A full explanation would require a deep understanding of precisely how the simulators function. Future work should reconcile the theoretical scaling expectations with the observed scaling behaviour. Regardless, given the fitting functions I can now extrapolate to higher qubit numbers, and this is shown in figure 3.7. From the figure, we can see the predicted cross over point occurring around 63 qubits for the GHZ circuit, perhaps within the simulation capacity of modern supercomputers. However, I

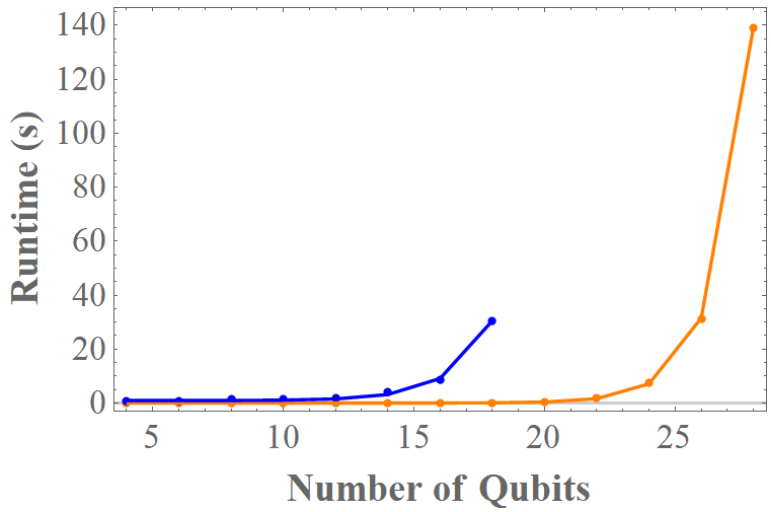


Figure 3.6: Simulation runtime is plotted against qubit number for both Aer (orange) and ExaTN-MPS (blue) when simulating the GHZ circuit. Fitting functions (curves) are plotted over simulation data (points).

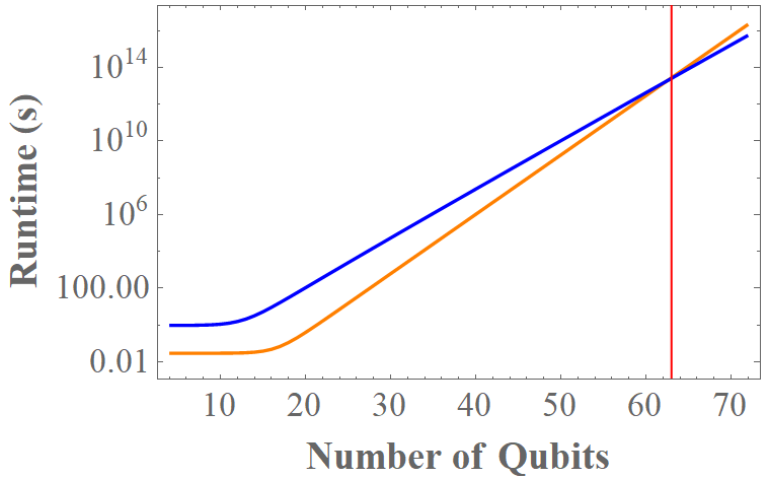


Figure 3.7: The runtime plot shown in figure 3.6 is extrapolated to higher qubit numbers for both Aer (orange) and ExaTN-MPS (blue) when simulating the GHZ circuit. The cross over point is indicated with a vertical, red line. Note the y-axis is on a log scale.

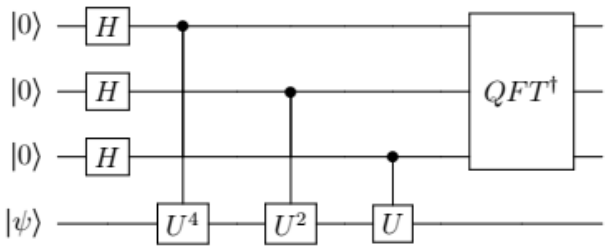


Figure 3.8: The quantum phase estimation circuit. Given unitary operation U and an eigenvector $|\psi\rangle$, this circuit is used to estimate the phase of the corresponding eigenvalue. The 3-qubit gate $QFT^\dagger$ denotes the conjugate-transpose of the 3-qubit quantum Fourier transform circuit.

am extrapolating a long way from the simulation data, so the precise cross over point has a large uncertainty.

3.2.2 Quantum Phase Estimation

Quantum phase estimation (QPE) is one of the most important circuits in quantum computing. It forms a key part of many quantum algorithms, including Shor's algorithm [6]. Given a unitary operator U and an eigenvector $|\psi\rangle$ such that

$$U|\psi\rangle = e^{2\pi i\theta}|\psi\rangle, \tag{3.12}$$

the QPE algorithm is used to estimate θ. A detailed description of how the QPE algorithm does this can be found in [25]. Note that the QPE circuit requires t qubits to estimate θ in addition to the qubits required to encode the eigenvector $|\psi\rangle$. The case for $t = 3$ is shown in figure 3.8. As a trivial example, consider

$$U = \begin{pmatrix} 1 & 0 \\ 0 & e^{2\pi i\theta} \end{pmatrix}, \tag{3.13}$$

which has an eigenvector

$$|\psi\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}. \tag{3.14}$$

Simulator	Expected Scaling	Fitting Function	a	b	c
Aer	$\mathcal{O}(2^{2n})$	$a + b2^{cn}$	0.078721	5.4331×10^{-7}	1.0842
ExaTN-MPS	$\mathcal{O}(n2^{3n/2})$	$a + bn2^{cn}$	65.877	0.15818	0.49444

Table 3.2: Fitting functions and parameters for the scaling behaviour of the Aer and ExaTN-MPS simulators for the QPE circuit. These are plotted in figure 3.9.

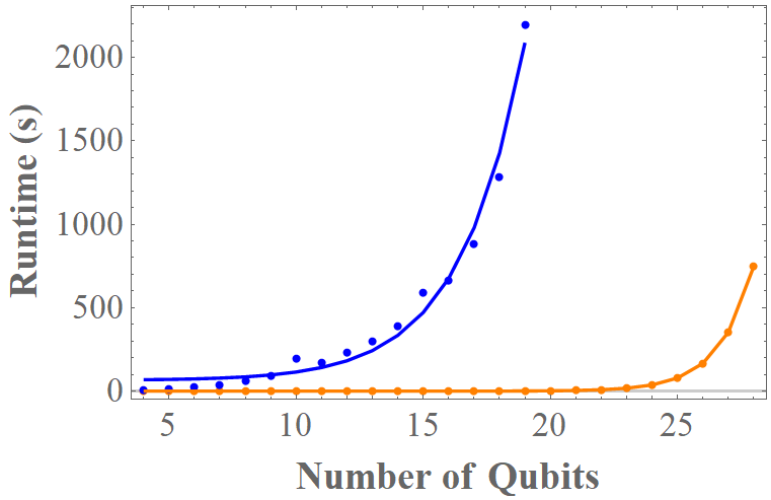


Figure 3.9: Simulation runtime is plotted against qubit number for both Aer (orange) and ExaTN-MPS (blue) when simulating the QPE circuit. Fitting functions (curves) are plotted over simulation data (points).

By choosing $\theta = 1/2^{n-1}$ for $n > 2$, the QPE algorithm is able to find the phase precisely using n qubits. This is the circuit I simulate with Aer and ExaTN-MPS for various qubit numbers n. The same error in ExaTN-MPS was present for this comparison, restricting n to be less than 20. Hence I once again find fitting functions whose parameters are given in table 3.2. The runtime comparison along with fitting functions is shown in figure 3.9. Similar to above, the fitted scaling behaviour is better than theoretically predicted. The extrapolation to higher qubit numbers is shown in figure 3.10. From the figure, we observe the predicted cross over occurring at around 40 qubits for the QPE circuit, this time certainly within the simulation capacity of modern supercomputers. However, as with the GHZ case, I have had to extrapolate very far from the simulation data.

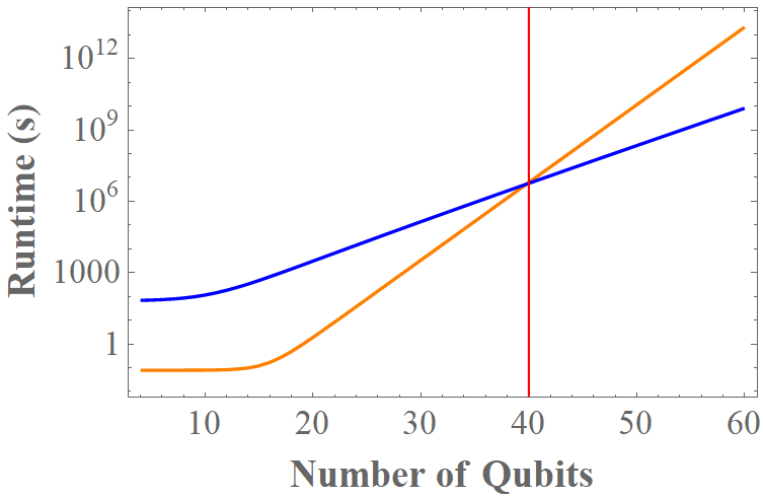


Figure 3.10: The runtime plot shown in figure 3.9 is extrapolated to higher qubit numbers for both Aer (orange) and ExaTN-MPS (blue) when simulating the QPE circuit. The cross over point is indicated with a vertical, red line. Note the y-axis is on a log scale.

So, while the existence of these cross over points is guaranteed by large n scaling behaviour, the exact location for particular circuits requires further investigation. Future work should involve using supercomputing resources to verify whether the cross over points occurs at the places I've identified. Verifying the existence of the cross over points within the simulation capacity of supercomputers would support the use of MPS simulations in general quantum computing research, particularly for investigating applications of NISQ devices with between 50 and 100 qubits.

3.3 Noisy Simulations

As mentioned in section 2.4.3, the most computationally efficient way to include noise in density matrix simulations is using a Kraus model. That is, we evolve the density matrix of the system through the circuit,

$$\rho \mapsto U\rho U^{\dagger}. \tag{3.15}$$

Then, at every point in the circuit you wish to introduce noise, you apply a Kraus channel,

$$\rho \mapsto \sum_i K_i \rho K_i^\dagger. \tag{3.16}$$

An alternative to using a density matrix simulation is to run a state-vector simulation many times, probabilistically introducing "noise gates" at appropriate places in the circuit, and taking the average over the outputs. This Monte-Carlo approach may often be preferred due to the savings in memory. Furthermore, depending on the number of repetitions performed, it may have a lower computational cost than the density matrix simulation. By doing this, noise can be introduced into Schrodinger simulations and MPS simulations. Another advantage of using a Kraus model to include noise in simulations is that it can give us an easy way to determine the noise gates we should use in the circuit and the probabilities with which to include them. Suppose that each of the K_i in the Kraus model can be written in the form

$$K_i = \sqrt{p_i}\tilde{K}_i, \tag{3.17}$$

where p_i is a non-negative, real constant and $\tilde{K}_i$ is unitary. Then the operators $\tilde{K}_i$ are the desired noise gates and the p_i give the probability distribution with which to include in the circuit. For example, the depolarising channel is a commonly used phenomenological noise model and can be written as

$$\rho \mapsto (1-p_0)\rho + \frac{p_0}{3}X\rho X + \frac{p_0}{3}Y\rho Y + \frac{p_0}{3}Z\rho Z, \tag{3.18}$$

where $p_0 \in [0,1]$ and X, Y, Z are the usual Pauli operators. In the Monte-Carlo version of this example, we insert the identity gate with probability $1-p_0$, and we insert X, Y, or Z with probability $\frac{p_0}{3}$. Therefore, given a Kraus model for noise in a quantum computer, we can incorporate noise into our preferred simulation method. Hence, the results from this chapter are also relevant to noisy

simulations.

With the above in mind, to perform noisy simulations of diamond quantum computers, it suffices to have an efficient state-vector simulator and a Kraus model for noise. In this chapter, I have shown that Schrodinger simulations are the most efficient widely-used simulation method, at least for small numbers of qubits. So, all that remains is to obtain a Kraus model for noise. There are two ways to obtain a Kraus model. The first way is to fit an empirical model (such as the above depolarising channel) to experimental data. As previously discussed, this yields a noise model that is non-generalisable and has no connection to the underlying noise. Therefore, I will focus on the second method of obtaining a Kraus model: deriving the model from the underlying sources of noise. In the following chapter, I outline a novel method for deriving a Kraus model for some noise sources. Then, in chapter 5 I verify the derived Kraus model.

Chapter 4

A Noise Model for Diamond Quantum Computers

Noisy simulations of quantum computers generally use phenomenological noise models. These models may be accurate for particular training circuits or hardware, but tend to poorly reflect the performance of a physical quantum computer in general. This makes these noise models inappropriate for investigating new quantum algorithms. Furthermore, the errors introduced by these noise models have no connection to physical noise in the quantum hardware. Hence, they cannot reliably provide feedback on the most important sources of noise to guide improvements in the hardware. In this chapter, I aim to overcome these issues by deriving a noise model for diamond quantum computers that takes into account the underlying physics. I use the Kraus representation because it is efficient and simple to include in a wide range of simulation methods, including those studied in the previous chapter.

In this model, I consider only nuclear spin qubits in NV-centres of diamond. I incorporate two main sources of noise that contribute to errors in the operation of quantum computers. First, interactions between the quantum computer and its environment cause qubit decoherence and relaxation. This will be investigated in section 4.1. Second, noise in the amplitude, phase, and frequency of radio-frequency (RF) pulses leads to imperfect single-qubit gate operations. In section 4.2 I present a novel method of deriving a Kraus model for noisy gates.

The method presented can be generalised to other sources of noise, including multi-qubit gate operations, and any other gate-based quantum computing architectures. I do not consider errors associated with state preparation and measurement as these are well-described phenomenologically [23].

4.1 Interactions with the Environment

The nuclear spin qubit in the diamond's NV-centre can occupy a ground state $|0\rangle$ or an excited state $|1\rangle$. Since diamond quantum computers are operated at room temperature, energy can be transferred to or from the environment in many ways such as electromagnetic interactions, and phonons in the diamond lattice. This energy transfer can cause the qubit to transition between the ground state and the excited state. The rate at which this process occurs is characterised by the T_1 relaxation time. Consider a qubit in the excited state, $|1\rangle$. Then the probability that it decays to the ground state, $|0\rangle$ in time t is given by

$$P_{relax} = 1 - e^{-t/T_1}. \tag{4.1}$$

This is called relaxation. The qubits may also experience decoherence. For nuclear spin qubits in diamond, this process is dominated a hyperfine coupling to nearby electron spins. As the electron relaxes and drops to the ground state, the nuclear qubit picks up a phase dependent on the strength of the coupling, causing decoherence. The rate of this process is characterised by the T_2 time. The electron and nuclear qubit are strongly coupled, however, in practice, we control the electron state to always be in either the ground state or the excited state. Therefore, no entanglement is created between the electron and the qubit and we are able to treat them separately. Then, since the qubit decoherence is

governed by the electron relaxation, we have the effective model,

$$T_2 \approx T_{1,e}, \tag{4.2}$$

where $T_{1,e}$ is the characteristic relaxation time for the electron spin. Note that what I am calling T_2 is generally referred to as *pure* T_2 in the literature, however the distinction is merely notational and does not affect the derivation. Given these noise mechanisms, we can write down the standard master equation for nuclear spin qubits undergoing interactions with the environment,

$$\dot{\rho} = \frac{1}{T_2} Z\rho Z + \frac{1}{2T_1}(\sigma_+\rho\sigma_+^\dagger + \sigma_-\rho\sigma_-^\dagger - \frac{1}{2}\{\sigma_+^\dagger\sigma_+, \rho\} - \frac{1}{2}\{\sigma_-^\dagger\sigma_-, \rho\}), \tag{4.3}$$

where Z is the Pauli Z operator, and $\sigma_\pm$ are linear combinations of the Pauli X and Y operators,

$$\sigma_\pm = X \mp iY. \tag{4.4}$$

Observe that equation (4.4) is in Lindblad form, with Hamiltonian $H = 0$, and Lindblad operators

$$\gamma_1 L_1 = \frac{1}{T_2} Z, \tag{4.5}$$

$$\gamma_2 L_2 = \frac{1}{2T_1}\sigma_+, \tag{4.6}$$

$$\gamma_3 L_3 = \frac{1}{2T_1}\sigma_-. \tag{4.7}$$

Now, I convert the master equation into the Kraus representation. If I assume a small evolution time t, then using equation (2.14) I obtain a one to one correspondence between Lindblad operators and non-identity Kraus operators given by

$$\tilde{K}_i = \sqrt{\gamma_i} L_i \sqrt{t}. \tag{4.8}$$

So, in this case we get

$$\tilde{K_1} = \sqrt{\frac{t}{T_2}}Z, \tag{4.9}$$

$$\tilde{K_2} = \sqrt{\frac{t}{2T_1}}\sigma_+, \tag{4.10}$$

$$\tilde{K_3} = \sqrt{\frac{t}{2T_1}}\sigma_-. \tag{4.11}$$

Then, to complete the Kraus model we include

$$\tilde{K_0} = \gamma_0 I, \tag{4.12}$$

where I is the identity operator and γ_0 is a constant such that the representation is normalised. That is,

$$\gamma_0 = 1 - \frac{t}{T_1} - \frac{t}{T_2}. \tag{4.13}$$

So, we have

$$\rho_{out} = \frac{t}{T_2}Z\rho_{in}Z + \frac{t}{2T_1}(\sigma_+\rho_{in}\sigma_+^\dagger + \sigma_-\rho_{in}\sigma_-^\dagger) + \left(1 - \frac{t}{T_1} - \frac{t}{T_2}\right)\rho_{in}. \tag{4.14}$$

Whilst the small-time approximation is generally valid in diamond quantum computers, it is an approximation we don't need to make. Instead, first, we rewrite the master equation as a matrix differential equation of the form

$$\dot{\rho} = (\mathcal{H} + \mathcal{G})\rho, \tag{4.15}$$

where ρ is now understood to be the vectorised density matrix, and

$$\mathcal{H} = -i(H \otimes I - I \otimes H), \tag{4.16}$$

$$\mathcal{G} = \sum_i \bar{L}_i \otimes L_i - \frac{1}{2}I \otimes (L_i^\dagger L_i) - \frac{1}{2}(\bar{L}_i^\dagger \bar{L}_i) \otimes I, \tag{4.17}$$

where the overline denotes complex conjugation. Since $H = 0$ we have $\mathcal{H} = 0$. Substituting the matrix form of the Lindblad operators into equation (4.18), we get

$$\mathcal{G} = \begin{pmatrix} -\frac{1}{T_1} & 0 & 0 & \frac{1}{T_1} \\ 0 & -\frac{1}{T_1} - \frac{1}{T_2} & 0 & 0 \\ 0 & 0 & -\frac{1}{T_1} - \frac{1}{T_2} & 0 \\ \frac{1}{T_1} & 0 & 0 & -\frac{1}{T_1} \end{pmatrix}. \tag{4.18}$$

Now that we have

$$\dot{\rho} = \mathcal{G}\rho, \tag{4.19}$$

we can solve the differential equation in the standard way to get

$$\rho_{out} = e^{\mathcal{G}t}\rho_{in}. \tag{4.20}$$

Taking the matrix exponential of $\mathcal{G}t$ yields

$$e^{\mathcal{G}t} = \begin{pmatrix} \frac{1+e^{-\frac{t}{T_1}}}{2} & 0 & 0 & \frac{1-e^{-\frac{t}{T_1}}}{2} \\ 0 & e^{-\frac{t}{2T_1}-\frac{2t}{T_2}} & 0 & 0 \\ 0 & 0 & e^{-\frac{t}{2T_1}-\frac{2t}{T_2}} & 0 \\ \frac{1-e^{-\frac{t}{T_1}}}{2} & 0 & 0 & \frac{1-e^{-\frac{t}{T_1}}}{2} \end{pmatrix}. \tag{4.21}$$

This matrix is referred to as the evolution matrix. The Choi matrix, χ, is obtained from the evolution matrix by a permutation of its elements,

$$\chi = \begin{pmatrix} \frac{1+e^{-\frac{t}{T_1}}}{2} & 0 & 0 & e^{-\frac{t}{2T_1}-\frac{2t}{T_2}} \\ 0 & \frac{1-e^{-\frac{t}{T_1}}}{2} & 0 & 0 \\ 0 & 0 & \frac{1-e^{-\frac{t}{T_1}}}{2} & 0 \\ e^{-\frac{t}{2T_1}-\frac{2t}{T_2}} & 0 & 0 & \frac{1-e^{-\frac{t}{T_1}}}{2} \end{pmatrix}. \tag{4.22}$$

Finally, we can convert this Choi matrix into a Kraus representation using the Choi-Jamiolkowski isomorphism described in chapter 2.3. I start by constructing

the matrix

$$e = U_\chi D_\chi^{\frac{1}{2}}, \tag{4.23}$$

where U_χ is a unitary matrix, and D_χ is a diagonal matrix such that

$$\chi = U_\chi D_\chi U_\chi^\dagger \tag{4.24}$$

is the usual eigen-decomposition of χ. The columns of e are then vectorised Kraus operators. Returning them to matrix form we obtain

$$K_0 = \sqrt{\frac{1}{4} + \frac{1}{4}e^{-\frac{t}{T_1}} + \frac{1}{2}e^{-\frac{t}{2T_1}-\frac{2t}{T_2}}}\, I, \tag{4.25}$$

$$K_1 = \sqrt{\frac{1}{4} + \frac{1}{4}e^{-\frac{t}{T_1}} - \frac{1}{2}e^{-\frac{t}{2T_1}-\frac{2t}{T_2}}}\, Z, \tag{4.26}$$

$$K_2 = \sqrt{\frac{1}{2} - \frac{1}{2}e^{-\frac{t}{T_1}}}\, \sigma_+, \tag{4.27}$$

$$K_3 = \sqrt{\frac{1}{2} - \frac{1}{2}e^{-\frac{t}{T_1}}}\, \sigma_-. \tag{4.28}$$

Thus, the Kraus representation for the part of my noise model describing system-environment interactions is

$$\rho_{out} = \sum_i K_i \rho_{in} K_i^\dagger, \tag{4.29}$$

and it is simple to check that making a small time approximation reduces this equation to equation (4.15).

4.2 Control Errors

4.2.1 Noisy Rotations

In diamond quantum computers, single-qubit gates are achieved by applying a RF pulse to the target qubit that causes a rotation on the Bloch sphere. Given a

pulse with amplitude Ω, phase ϕ, frequency Δ, and duration t, it can be shown [52] that the Hamiltonian acting on the qubit is

$$H = \frac{1}{2}(\Omega\cos\phi, \Omega\sin\phi, \Delta)\cdot\vec{\sigma}, \tag{4.30}$$

where $\vec{\sigma}$ is the vector of Pauli matrices. Then the evolution of the qubit is

$$U = e^{-iHt} = e^{-\frac{i}{2}\vec{R}\cdot\vec{\sigma}}, \tag{4.31}$$

where the qubit is rotated about the axis

$$\vec{R} = (\Omega t\cos\phi, \Omega t\sin\phi, \Delta t), \tag{4.32}$$

and the angle of rotation, γ, is encoded in the magnitude of the vector $\vec{R}$,

$$\gamma = |\vec{R}| = \sqrt{\Omega^2 + \Delta^2}t. \tag{4.33}$$

I will denote a particular RF pulse by the quadruple $(\Omega, \phi, \Delta, t)$. By an appropriate choice of parameters, we can perform any rotation of the Bloch sphere and hence any single-qubit gate. A perfect gate operation is given by the evolution

$$\rho_{out} = U^\dagger\rho_{in}U. \tag{4.34}$$

However, we do not have perfect control over the parameters of the RF pulse. Due to various sources of classical and quantum noise, the actual pulse will be $(\Omega + \delta\Omega, \phi + \delta\phi, \Delta + \delta\Delta, t + \delta t)$. However, we have good control over the pulse duration compared with the other variables, so therefore we take $\delta t = 0$. The

noisy rotation vector is then

$$\begin{aligned}\vec{R}' &= ((\Omega+\delta\Omega)t\cos(\phi+\delta\phi),(\Omega+\delta\Omega t\sin(\phi+\delta\phi),(\Delta+\delta\Delta)t)\\ &= \vec{R}+(\delta\Omega t(\cos\phi-\delta\phi\sin\phi)-\Omega t\delta\phi\sin\phi,\Omega t\delta\phi\cos\phi+\delta\Omega t(\sin\phi+\delta\phi\cos\phi,\delta\Delta t)\\ &= \vec{R}+\delta\vec{R},\end{aligned} \tag{4.35}$$

where I have made the small angle approximations $\cos\delta\phi = 1$ and $\sin\delta\phi = \delta\phi$. The noisy gate operation is then

$$U' = e^{-\frac{i}{2}(\vec{R}+\delta\vec{R})\cdot\vec{\sigma}}, \tag{4.36}$$

where I denote the presence of noise with a prime. My goal now is to derive a Kraus representation for the noisy single-qubit gate. To do this, first suppose we could decompose U' into an ideal part and a noisy part,

$$U' = U\xi, \tag{4.37}$$

where U is the ideal rotation defined in equation (4.31), and ξ is some operator describing the noisy processes. Then we could write

$$\rho'_{out} = U'^{\dagger}\rho_{in}U' = \xi^{\dagger}U^{\dagger}\rho_{in}U\xi = \xi^{\dagger}\rho_{out}\xi, \tag{4.38}$$

and go on to derive a Kraus model. But first we must find the decomposition in equation (4.37).

4.2.2 Deriving the Noise Operator

Since $\vec{R}\cdot\vec{\sigma}$ does not commute with $\delta\vec{R}\cdot\vec{\sigma}$, decomposing equation 4.37 is not straightforward. Consider the explicit description of the Zassenhaus formula [53]

$$e^{A+B} = e^{A}\left\{1 + \sum_{p=0}^{\infty}\sum_{n_1,\ldots n_p=1}^{\infty} \frac{(-1)^{n_p+\ldots+n_1-p} n_p \ldots n_1}{n_p(n_p+n_{p-1})\ldots(n_p+\ldots+n_1)} \mathcal{B}_{n_p}\ldots\mathcal{B}_{n_1}\right\}, \quad (4.39)$$

where A and B are possibly non-commuting operators, and $\mathcal{B}_m$ is defined to be

$$\mathcal{B}_m \equiv \frac{1}{m!}(\mathcal{L}_A)^{m-1}B, \quad (4.40)$$

where

$$\mathcal{L}_A B \equiv [A, B]. \quad (4.41)$$

This expression is a decomposition of e^{A+B} in orders of B. That is, truncating the expression after $p = 1$ gives the decomposition to first order in B, truncating after $p = 2$ gives the decomposition to second order in B, and so on. So, if we assume the erroneous part of the rotation, $\delta\vec{R}$, is small, we can approximate ξ to first order in $\delta\vec{R}\cdot\vec{\sigma}$. Therefore, using only up to the $p = 1$ term in equation (4.39), we have

$$\begin{aligned} e^{-\frac{i}{2}(\vec{R}+\delta\vec{R})\cdot\vec{\sigma}} &= e^{-i\vec{R}\cdot\vec{\sigma}}\left\{1 + \sum_{n=1}^{\infty}(-1)^{n-1}\mathcal{B}_n\right\} \\ &= e^{-\frac{i}{2}\vec{R}\cdot\vec{\sigma}}\left\{1 + \sum_{n=1}^{\infty}\frac{(-1)^{n-1}}{n!}(\mathcal{L}_{-\frac{i}{2}\vec{R}\cdot\vec{\sigma}})^{n-1}\left(-\frac{i}{2}\delta\vec{R}\cdot\vec{\sigma}\right)\right\}. \end{aligned} \quad (4.42)$$

To proceed we need the following proposition.

Proposition 1. *With everything as defined in the text,*

$$(\mathcal{L}_{-\frac{i}{2}\vec{R}\cdot\vec{\sigma}})^k\left(-\frac{i}{2}\delta\vec{R}\cdot\vec{\sigma}\right) = \begin{cases} (-\frac{1}{2}\gamma^2)^{\frac{k-1}{2}}\mathcal{L}_{-\frac{i}{2}\vec{R}\cdot\vec{\sigma}}\left(-\frac{i}{2}\delta\vec{R}\cdot\vec{\sigma}\right) & k \geq 1 \text{ is odd} \\ (-\frac{1}{2}\gamma^2)^{\frac{k}{2}-1}(\mathcal{L}_{-\frac{i}{2}\vec{R}\cdot\vec{\sigma}})^2\left(-\frac{i}{2}\delta\vec{R}\cdot\vec{\sigma}\right) & k \geq 2 \text{ is even} \end{cases}.$$

Proof. The proof is a simple case of mathematical induction, and is deferred to Appendix A. □

To use the proposition, we first split the sum in equation (4.42) into three pieces corresponding to $n-1=0$, $n-1$ odd, and $n-1$ even,

$$e^{-\frac{i}{2}(\vec{R}+\delta\vec{R})\cdot\vec{\sigma}} = e^{-\frac{i}{2}\vec{R}\cdot\vec{\sigma}}\Bigg\{1 - \frac{i}{2}\delta\vec{R}\cdot\vec{\sigma} + \sum_{n \text{ odd}} \frac{(-1)^n}{(n+1)!}(\mathcal{L}_{-\frac{i}{2}\vec{R}\cdot\vec{\sigma}})^n\left(-\frac{i}{2}\delta\vec{R}\cdot\vec{\sigma}\right) + \sum_{n>0 \text{ even}} \frac{(-1)^n}{(n+1)!}(\mathcal{L}_{-\frac{i}{2}\vec{R}\cdot\vec{\sigma}})^n\left(-\frac{i}{2}\delta\vec{R}\cdot\vec{\sigma}\right)\Bigg\}. \tag{4.43}$$

Then substitute the result of Proposition 1,

$$e^{-\frac{i}{2}(\vec{R}+\delta\vec{R})\cdot\vec{\sigma}} = e^{-\frac{i}{2}\vec{R}\cdot\vec{\sigma}}\Bigg\{1 - \frac{i}{2}\delta\vec{R}\cdot\vec{\sigma} + \sum_{n \text{ odd}} \frac{(-1)^n}{(n+1)!}\left(-\frac{1}{2}\gamma^2\right)^{n-1}\mathcal{L}_{-\frac{i}{2}\vec{R}\cdot\vec{\sigma}}\left(-\frac{i}{2}\delta\vec{R}\cdot\vec{\sigma}\right) + \sum_{n>0 \text{ even}} \frac{(-1)^n}{(n+1)!}\left(-\frac{1}{2}\gamma^2\right)^{k-1}(\mathcal{L}_{-\frac{i}{2}\vec{R}\cdot\vec{\sigma}})^2\left(-\frac{i}{2}\delta\vec{R}\cdot\vec{\sigma}\right)\Bigg\}. \tag{4.44}$$

The sums over odd and even terms converge and are given by hypergeometric functions. I will denote them $F(\gamma)$ and $G(\gamma)$ respectively,

$$\sum_{n \text{ odd}} \frac{(-1)^n}{(n+1)!}\left(-\frac{1}{2}\gamma^2\right)^{n-1} \equiv F(\gamma), \tag{4.45}$$

$$\sum_{n \text{ even}} \frac{(-1)^n}{(n+1)!}\left(-\frac{1}{2}\gamma^2\right)^{n-1} \equiv G(\gamma). \tag{4.46}$$

Therefore we get,

$$e^{-\frac{i}{2}(\vec{R}+\delta\vec{R})\cdot\vec{\sigma}} = e^{-\frac{i}{2}\vec{R}\cdot\vec{\sigma}}\Bigg\{1 + \frac{1}{4}(iG(\gamma)\gamma^2 - 2i)\delta\vec{R}\cdot\vec{\sigma} + \frac{F(\gamma)}{4}(\delta\vec{R}\cdot\vec{\sigma})(\vec{R}\cdot\vec{\sigma}) - \frac{F(\gamma)}{4}(\vec{R}\cdot\vec{\sigma})(\delta\vec{R}\cdot\vec{\sigma}) - \frac{iG(\gamma)}{4}(\vec{R}\cdot\vec{\sigma})(\delta\vec{R}\cdot\vec{\sigma})(\vec{R}\cdot\vec{\sigma})\Bigg\}, \tag{4.47}$$

where we have also expanded out $\mathcal{L}_{-\frac{i}{2}\vec{R}\cdot\vec{\sigma}}(-\frac{i}{2}\delta\vec{R}\cdot\vec{\sigma})$ and $(\mathcal{L}_{-\frac{i}{2}\vec{R}\cdot\vec{\sigma}})^2(-\frac{i}{2}\delta\vec{R}\cdot\vec{\sigma})$.

Hence, we finally have the noise operator,

$$\begin{aligned}\xi = 1 + \frac{1}{4}(iG(\gamma)\gamma^2 - 2i)\delta\vec{R}\cdot\vec{\sigma} + \frac{F(\gamma)}{4}(\delta\vec{R}\cdot\vec{\sigma})(\vec{R}\cdot\vec{\sigma}) \\ - \frac{F(\gamma)}{4}(\vec{R}\cdot\vec{\sigma})(\delta\vec{R}\cdot\vec{\sigma}) - \frac{iG(\gamma)}{4}(\vec{R}\cdot\vec{\sigma})(\delta\vec{R}\cdot\vec{\sigma})(\vec{R}\cdot\vec{\sigma}).\end{aligned} \tag{4.48}$$

4.2.3 Deriving the Kraus Model

Now we can work towards a Kraus model for the noisy operation. Recall we have

$$\rho'_{out} = \xi^\dagger \rho_{out} \xi. \tag{4.49}$$

If we expand out the right hand side we get a large number of terms involving the noisy part of the rotation $\delta\vec{R}\cdot\vec{\sigma}$, which depends on the random variables $(\delta\Omega, \delta\phi, \delta\Delta)$ as in equation (4.35). To get a Kraus model, we want to integrate over these random variables to get the average output,

$$\bar{\rho}'_{out} = \iiint d\delta\Omega d\delta\phi d\delta\Delta P(\delta\Omega, \delta\phi, \delta\Delta)\rho'_{out}, \tag{4.50}$$

where P is the probability distribution function for $(\delta\Omega, \delta\phi, \delta\Delta)$. Many noise processes obey a normal distribution due to the central limit theorem. Hence I assume P to be the multivariate normal distribution. For real electronics, there will likely be some correlations between the parameters, however I assume these are small enough to ignore. Therefore, the covariance matrix is diagonal and P is given by

$$P(\delta\Omega, \delta\phi, \delta\Delta) = \frac{1}{(2\pi)^{3/2}} \frac{1}{\sigma_\Omega \sigma_\phi \sigma_\Delta} e^{-\frac{1}{2}\left(\frac{\delta\Omega^2}{\sigma_\Omega^2} + \frac{\delta\phi^2}{\sigma_\phi^2} + \frac{\delta\Delta^2}{\sigma_\Delta^2}\right)}. \tag{4.51}$$

Due to this choice of distribution, the only terms in ρ'_{out} that survive the integration are those that contain $\delta\Omega^2$, $\delta\phi^2$, or $\delta\Delta^2$. All linear terms and all cross terms will average to 0. As a demonstration, observe that one term in the expansion of

ρ'_{out} is

$$(\delta\vec{R}\cdot\vec{\sigma})\rho_{out}(\delta\vec{R}\cdot\vec{\sigma}), \tag{4.52}$$

where we ignore any constants for now. Taking the average with respect to P yields,

$$\begin{aligned}&\sigma_\Omega^2(t\cos\phi X + t\sin\phi Y)\rho_{out}(t\cos\phi X + t\sin\phi Y)\\&+\sigma_\phi^2(\Omega t\sin\phi X - \Omega t\cos\phi Y)\rho_{out}(\Omega t\sin\phi X - \Omega t\cos\phi Y) + \sigma_\Delta^2(tZ)\rho_{out}(tZ).\end{aligned} \tag{4.53}$$

We will get very similar expressions when we integrate the other terms in ρ'_{out} so for convenience we define

$$K_1 = \sigma_\Omega t\cos\phi X + \sigma_\Omega t\sin\phi Y, \tag{4.54}$$

$$K_2 = \sigma_\phi\Omega t\sin\phi X - \sigma_\phi\Omega t\cos\phi Y, \tag{4.55}$$

$$K_3 = \sigma_\Delta tZ \tag{4.56}$$

so that

$$\iiint d\delta\Omega d\delta\phi d\delta\Delta P(\delta\Omega,\delta\phi,\delta\Delta)(\delta\vec{R}\cdot\vec{\sigma})\rho_{out}(\delta\vec{R}\cdot\vec{\sigma}) = \sum_i K_i\rho_{out}K_i, \tag{4.57}$$

and we see that this term directly yields a Kraus representation since $K_i^\dagger = K_i$. If we repeat the integration for the rest of the terms we obtain the following

expression,

$$\begin{aligned}\bar{\rho}'_{out} = \rho_{out} &+ \frac{F(\gamma)^2}{16}(\vec{R}\cdot\vec{\sigma})\sum_i K_i\rho_{out}K_i(\vec{R}\cdot\vec{\sigma}) \\ &+ \frac{F(\gamma)^2}{16}\left(\sum_i K_i(\vec{R}\cdot\vec{\sigma})\rho_{out}(\vec{R}\cdot\vec{\sigma})K_i\right) + \frac{(2-G(\gamma)\gamma^2)}{16}\left(\sum_i K_i\rho_{out}K_i\right) \\ &+ \frac{G(\gamma)^2}{16}\left(\sum_i (\vec{R}\cdot\vec{\sigma})K_i(\vec{R}\cdot\vec{\sigma})\rho_{out}(\vec{R}\cdot\vec{\sigma})K_i(\vec{R}\cdot\vec{\sigma})\right) \\ &- \frac{F(\gamma)^2}{16}\left(\sum_i (\vec{R}\cdot\vec{\sigma})K_i\rho_{out}(\vec{R}\cdot\vec{\sigma})K_i + K_i(\vec{R}\cdot\vec{\sigma})\rho_{out}K_i(\vec{R}\cdot\vec{\sigma})\right) \\ &+ \frac{a(2-G(\gamma)\gamma^2)}{16}\Bigg(\sum_i K_i\rho_{out}K_i(i\vec{R}\cdot\vec{\sigma}) - K_i\rho_{out}(i\vec{R}\cdot\vec{\sigma})K_i \\ &\qquad + K_i(i\vec{R}\cdot\vec{\sigma})\rho_{out}K_i - (i\vec{R}\cdot\vec{\sigma})K_i\rho_{out}K_i\Bigg) \\ &+ \frac{G(\gamma)(2-G(\gamma)\gamma^2)}{16}\left(\sum_i K_i\rho_{out}(\vec{R}\cdot\vec{\sigma})K_i(\vec{R}\cdot\vec{\sigma}) + (\vec{R}\cdot\vec{\sigma})K_i(\vec{R}\cdot\vec{\sigma})\rho_{out}K_i\right) \\ &- \frac{F(\gamma)G(\gamma)}{16}\Bigg(\sum_i K_i(\vec{R}\cdot\vec{\sigma})\rho_{out}(\vec{R}\cdot\vec{\sigma})K_i(i\vec{R}\cdot\vec{\sigma}) - (\vec{R}\cdot\vec{\sigma})K_i\rho_{out}(i\vec{R}\cdot\vec{\sigma})K_i(\vec{R}\cdot\vec{\sigma}) \\ &\qquad + (\vec{R}\cdot\vec{\sigma})K_i(i\vec{R}\cdot\vec{\sigma})\rho_{out}K_i(\vec{R}\cdot\vec{\sigma}) - (i\vec{R}\cdot\vec{\sigma})K_i(\vec{R}\cdot\vec{\sigma})\rho_{out}(\vec{R}\cdot\vec{\sigma})K_i\Bigg)\end{aligned} \tag{4.58}$$

Observe that the first five terms in equation (4.58) are in the right form for a Kraus model. To convert the remaining four terms into the Kraus representation, we require the following proposition.

Proposition 2. *This proposition has four parts. In each part the right hand side is in the Kraus representation.*

a. *For any operator A,*

$$A\rho A + A^\dagger\rho A^\dagger = (A+A^\dagger)\rho(A+A^\dagger) - A\rho A^\dagger - A^\dagger\rho A.$$

b. *For operators A and B such that $A^\dagger = -A$ and $B^\dagger = B$,*

$$[B, A]\rho B + B\rho[B, A] = [A, B]\rho[A, B] + B\rho B - ([A, B] + B)\rho([A, B] + B).$$

c. *For operators A and B such that $A^\dagger = A$ and $B^\dagger = B$,*

$$A\rho BAB + BAB\rho A = (A + BAB)\rho(A + BAB) - A\rho A - BAB\rho BAB.$$

d. *For operators A and B such that $B^\dagger = -B$,*

$$AB\rho A^\dagger - A\rho BA^\dagger = (AB + A)\rho(A^\dagger - BA^\dagger) - A\rho A^\dagger + AB\rho BA^\dagger,$$

and

$$A^\dagger\rho AB - BA^\dagger\rho A = (A^\dagger - BA^\dagger)\rho(AB + A) - A^\dagger\rho A + BA^\dagger\rho AB.$$

Proof. The proof is simple algebra and is omitted. □

Proposition 2 shows we can rewrite the last four terms in equation (4.58) to yield a Kraus representation of the noisy gate operation,

$$\bar{\rho}'_{out} = \sum_{i=1}^{31} E_i \rho_{out} E_i^\dagger, \tag{4.59}$$

where the explicit descriptions of the Kraus operators E_i is given in Appendix B. Having 31 Kraus operators is cumbersome to deal with, and it is also unnecessary. The Choi-Jamiolkowski isomorphism tells us we need at most 4 Kraus operators to describe any single-qubit evolution. Using the methods outlined in section 2.3, if we transform (4.59) to a Choi matrix, then convert back to a Kraus representation, we will get 4 Kraus operators that are equivalent to the 31 derived above. The process is as follows:

1. Define all 31 Kraus operators E_i in terms of Ω, ϕ, Δ, t, σ_Ω, σ_ϕ, and σ_Δ as

given in Appendix B.

2. Define e_i to be the vectorised E_i and form the matrix $e = [e_1, ..., e_{31}]$.

3. Compute the Choi matrix $\chi = ee^\dagger$.

4. Diagonalise the Choi matrix $\chi = U_\chi D_\chi U_\chi^\dagger$.

5. Define the new matrix $e' = U_\chi D_\chi^{1/2} = [e'_1, e'_2, e'_3, e'_4]$ where the e'_i are vectorised Kraus operators.

6. Convert e'_i to matrix form to obtain 4 Kraus operators E'_i that are equivalent to the original 31.

7. The final step is to enforce the normalisation condition for Kraus operators, $\sum_i E_i'^\dagger E'_i = I$.

Performing this procedure in full generality to try to find an analytic solution is computationally intensive and does not finish in a reasonable amount of time. This isn't a concern because any quantum circuit of interest will comprise only a finite set of gates. Hence, the procedure outlined above can be used to find the Kraus representation for the particular noisy gates involved in the circuit. Selecting a particular gate U fixes the values of Ω, ϕ, Δ, and t. Then the Kraus representation for U' can be computed numerically using experimentally determined values for σ_Ω, σ_ϕ, and σ_Δ.

4.2.4 An Analytic Form

In practice, we are able to numerically compute the Kraus operators for a particular gate. However, it would still be of interest to find an analytic description. As mentioned above, we cannot do this in full generality due to computational constraints. However, recall that in diamond quantum computers, all single-qubit gates are performed as combinations of rotations about the x and y axes of the

Basis Element	Contribution to M
I	$\frac{1}{2}(M_{11} + M_{22})$
X	$\frac{1}{2}(M_{12} + M_{21})$
Y	$\frac{i}{2}(M_{12} - M_{21})$
Z	$\frac{1}{2}(M_{11} - M_{22})$

Table 4.1: The contribution of each basis element $\{I, X, Y, Z\}$ to the 2×2 hermitian matrix M

Bloch sphere. Therefore, consider first an x-axis rotation. This can be written as the rotation $(\gamma, 0, 0, 1.0)$ where γ is the angle of rotation. I compute the Kraus operators for a range of values of $\gamma \in (0, \pi]$. Each γ yields four Kraus operators, denoted by K_1, K_2, K_3, and K_4. Importantly, since the Choi matrix is hermitian (by construction), the Kraus operators that we derive from it will also be hermitian. Therefore, since $\{I, X, Y, Z\}$ forms a basis for 2×2 hermitian matrices, we can decompose the derived K_i in terms of this basis set. The contribution from each basis element is summarised in table 4.1. Doing this for each γ, I can plot the decomposition of each Kraus operator K_i as a function of rotation angle γ. This is shown in figure 4.1. We observe that for x-axis rotations, the Kraus operators are an identity (K_1), an X operator (K_2), and two admixtures of Y and Z operators (K_3 and K_4). The amplitude of K_2 is negligible compared to the other Kraus operators so it can safely be ignored. To satisfy the normalisation condition for the Kraus representation, we require K_3 and K_4 to be of the form

$$\sqrt{k(\gamma)}\left(\cos\left(\theta(\gamma)\right)Y + \sin\left(\theta(\gamma)\right)Z\right). \tag{4.60}$$

Figure 4.2 shows $k(\gamma)$ and $\theta(\gamma)$ for K_3 and K_4 plotted with fitting functions. The fitting functions and their parameters are given in table 4.2. Hence, the Kraus

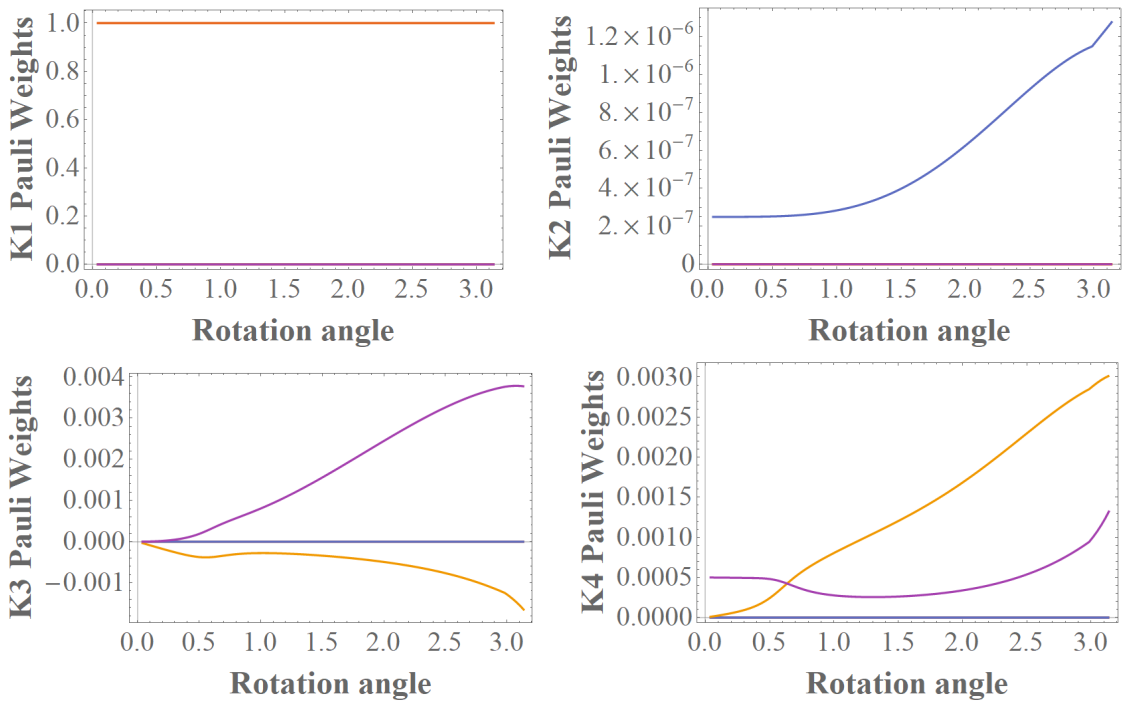


Figure 4.1: The decomposition of each Kraus operator K_i for $i \in \{1, 2, 3, 4\}$ in terms of I (orange), X (blue), Y (yellow), and Z (purple) is plotted against rotation angle about the x-axis of the Bloch sphere.

Data	Fitting Function	a	b	c	d
$k_3(\gamma)$	$a\gamma^6 + b\gamma^4 + c\gamma^2 + d$	-2.42365×10^{-8}	3.62016×10^{-7}	4.79505×10^{-7}	-2.22951×10^{-8}
$\theta_3(\gamma)$	$a + be^{c \tan^2(d\gamma)}$	-0.22537	-1.34625	-2.80912	0.78713
$k_4(\gamma)$	$a\gamma^6 + b\gamma^4 + c\gamma^2 + d$	-1.3614×10^{-9}	8.1533×10^{-8}	3.90500×10^{-7}	2.26028×10^{-7}
$\theta_4(\gamma)$	$a + be^{c \tan^2(d\gamma)}$	1.34543	-1.34625	-2.80912	0.78713

Table 4.2: Fitting functions and parameters for the curves plotted in figure 4.2.

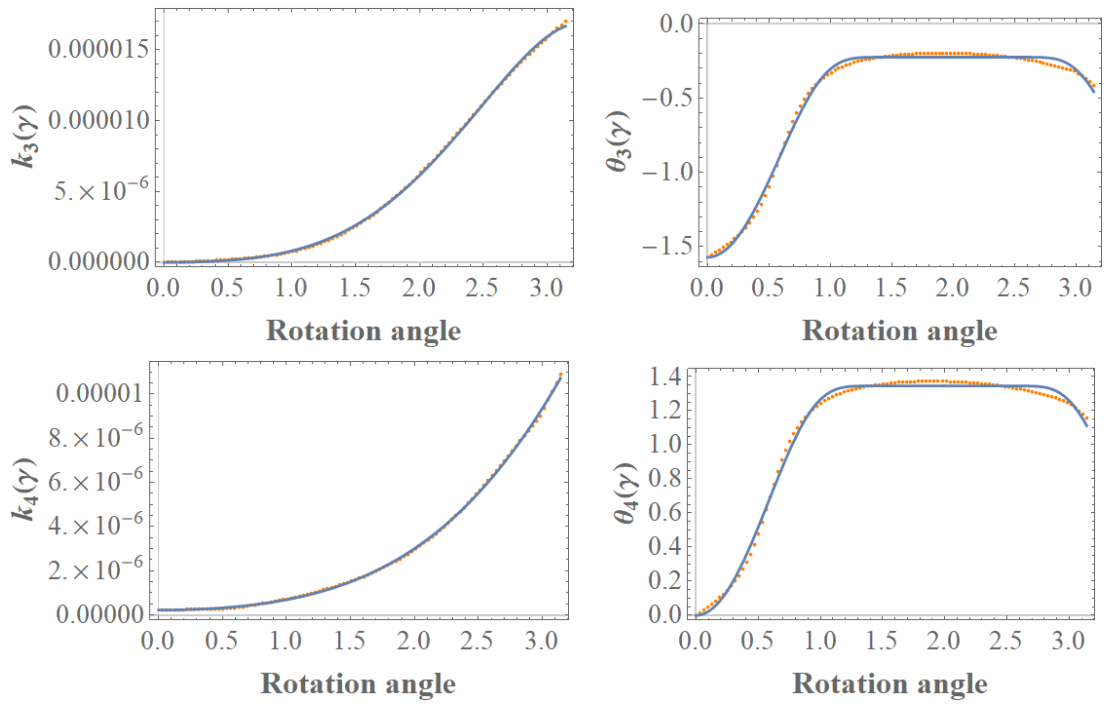


Figure 4.2: The derived amplitude and phase of K_3 and K_4 in the $Y-Z$ plane are plotted against the rotation angle about the x-axis on the Bloch sphere in orange. Functions are fitted to the data and shown in blue.

model for rotations about the x-axis by angle γ is given by

$$\begin{aligned}\rho'_{out} =&(1 - k_3(\gamma) - k_4(\gamma))\rho_{out} \\ &+ k_3(\gamma)\left(\cos\left(\theta_3(\gamma)\right)Y + \sin\left(\theta_3(\gamma)\right)Z\right)\rho_{out}\left(\cos\left(\theta_3(\gamma)\right)Y + \sin\left(\theta_3(\gamma)\right)Z\right) \\ &+ k_4(\gamma)\left(\cos\left(\theta_4(\gamma)\right)Y + \sin\left(\theta_4(\gamma)\right)Z\right)\rho_{out}\left(\cos\left(\theta_4(\gamma)\right)Y + \sin\left(\theta_4(\gamma)\right)Z\right).\end{aligned} \tag{4.61}$$

where ρ_{out} is the ideal output state. We can perform an entirely analogous procedure for rotations about the y-axis. However in this case, the Kraus operators are the identity, a negligible Y operator, and two admixtures of the X and Z operators. The normalisation condition for the Kraus representation implies we can write these admixture operators in the form

$$\sqrt{l(\gamma)}\left(\cos\left(\phi(\gamma)\right)X + \sin\left(\phi(\gamma)\right)Z\right). \tag{4.62}$$

I again extract $k(\gamma)$ and $\theta(\gamma)$ for each of the two admixture operators and fit functions to them. These fitting functions and their fitting parameters are given in table 4.3. Hence, the Kraus model for rotations about the y-axis by angle γ is given by

$$\begin{aligned}\rho'_{out} =&(1 - l_3(\gamma) - l_4(\gamma))\rho_{out} \\ &+ l_3(\gamma)\left(\cos\left(\phi_3(\gamma)\right)Y + \sin\left(\phi_3(\gamma)\right)Z\right)\rho_{out}\left(\cos\left(\phi_3(\gamma)\right)Y + \sin\left(\phi_3(\gamma)\right)Z\right) \\ &+ l_4(\gamma)\left(\cos\left(\phi_4(\gamma)\right)Y + \sin\left(\phi_4(\gamma)\right)Z\right)\rho_{out}\left(\cos\left(\phi_4(\gamma)\right)Y + \sin\left(\phi_4(\gamma)\right)Z\right).\end{aligned} \tag{4.63}$$

In this section I have defined a novel, completely analytic noise model for single-qubit gates in diamond quantum computers. Furthermore, I have provided a numerical algorithm to derive the Kraus operators for any given single-qubit gate. While I do not consider it in this thesis, the techniques used to derive the Kraus model for single-qubit gates can be used to derive a Kraus model for

Data	Fitting Function	a	b	c	d
$l_3(\gamma)$	$a\gamma^6 + b\gamma^4 + c\gamma^2 + d$	-1.36145×10^{-9}	8.15328×10^{-8}	3.904500×10^{-7}	2.26028×10^{-7}
$\phi_3(\gamma)$	$a + be^{c \tan^2(d\gamma)}$	-1.34543	1.34625	-2.80913	0.78713
$l_4(\gamma)$	$a\gamma^6 + b\gamma^4 + c\gamma^2 + d$	-2.42365×10^{-8}	3.62016×10^{-7}	4.79505×10^{-7}	-2.2295×10^{-8}
$\phi_4(\gamma)$	$a + be^{c \tan^2(d\gamma)}$	0.22537	1.34625	-2.80912	0.78713

Table 4.3: Fitting functions and parameters for the amplitude and phase in the $X - Z$ plane for Kraus operators corresponding to rotations about the y-axis on the Bloch sphere.

two-qubit gates.

In this chapter, I have shown how two of the main sources of noise in the operation of diamond quantum computers can be described as a Kraus model via a first-principles derivation. In chapter 5, I investigate validity and utility of the derived model.

Chapter 5

Validating the Noise Model

In chapter 3 I investigated state of the art methods for simulating noiseless quantum computers. I determined that for small scale simulations with few qubits, Schrodinger simulations are preferred in general. MPS simulations may begin to outperform Schrodinger simulations for simulations with greater than around 35 qubits for particular tasks. In chapter 4, I derived a Kraus operator noise model for diamond quantum computers that can be incorporated into existing simulators. In this chapter, I will investigate the validity and utility of the derived model.

In section 5.1, I investigate the validity of the noise model by performing noisy density matrix simulations of single qubit circuits and comparing the performance of the derived noise model with a standard depolarising channel. In section 5.2, I demonstrate an application of the derived noise model that is not possible with a phenomenological model.

5.1 Derived Model vs. Depolarising Channel

In the previous chapter, I first derived a Kraus model to describe decoherence. This model (in other forms) has been studied previously and is implemented in IBM's Aer simulator as a noise option, hence I don't consider it in this section. I then derived a novel noise model to describe control errors in single-qubit gates. To verify that this derived noise model provides an accurate description of reality,

we would ideally need data from a real diamond quantum computer to compare it to. We do not currently have such data, so we need an alternative benchmark. Since the noise model does not yet account for errors in multi-qubit gates, we only need to consider single-qubit circuits. Therefore, we can take a Monte Carlo approach to very accurately mimic the operation of a real quantum computer, where we only consider noise arising from control errors. Recall from the previous chapter that single-qubit gates are implemented by RF pulses that cause a rotation of the Bloch sphere,

$$U = e^{-\frac{i}{2}\vec{R}\cdot\vec{\sigma}}, \tag{5.1}$$

where $\vec{R}$ defines the rotation axis and $\gamma = |\vec{R}|$ defines the rotation angle. Consider a single-qubit quantum circuit $U_n U_{n-1} ... U_2 U_1$ where each U_i is in the form of equation (5.1). The procedure for the Monte Carlo simulation is as follows:

1. Initialise $\rho_{in} = |0\rangle\langle 0|$.
2. Determine the RF pulse $(\Omega, \phi, \Delta, t)$ that gives the rotation vector $\vec{R}$ associated with U_i, as in equation (4.32).
3. Randomly sample errors $(\delta\Omega, \delta\phi, \delta\Delta)$ from normal distributions with mean zero and variances $(\sigma_\Omega, \sigma_\phi, \sigma_\Delta)$.
4. Compute the noisy part of the rotation $\delta\vec{R}$ as in equation (4.35).
5. Compute the noisy operation $U_i' = e^{-\frac{i}{2}(\vec{R}+\delta\vec{R})\cdot\sigma}$.
6. Evolve the state $\rho \mapsto U_i' \rho U_i'^{\dagger}$.
7. Repeat steps 2-6 for each U_i to obtain the output state ρ_{out}.
8. Repeat steps 1-7 N times to produce a collection of outputs $\{\rho_{out,1}, ...\rho_{out,N}\}$.
9. Take the mean of the output density matrices to obtain the average output $\bar{\rho}_{out}$.

I will use the output of such Monte Carlo simulations as the benchmark for the derived noise model. However, we still need a performance metric. I will use the total variation distance (TVD), a statistical measure used to compare the similarity of probability distributions. We can define the TVD between two density matrices as

$$TVD(\rho_1, \rho_2) = \frac{1}{2} \sum_i | \langle i | \rho_1 | i \rangle - \langle i | \rho_2 | i \rangle |, \tag{5.2}$$

where the sum is taken over all basis states $|i\rangle$. In the case of single-qubit circuits $|i\rangle \in \{|0\rangle, |1\rangle\}$. The TVD is a popular choice in quantum computing research [23] because it compares only the measurable parts of the density matrices. The TVD is 0 when two density matrices are identical and 1 in the worst case where there is no correlation. Hence, we can define a more standard fidelity metric as

$$\mathcal{F}_{TVD}(\rho_1, \rho_2) = 1 - TVD(\rho_1, \rho_2), \tag{5.3}$$

where 1 is perfect and 0 is the worst case. I compare the performance of the derived noise model against a standard depolarising channel, as described in section 2.4.3, using Monte Carlo simulation results as the expected output of a "real" quantum computer. I will denote the depolarising channel by

$$\mathcal{D}[\rho] \equiv (1 - p_0)\rho + \frac{p_0}{3} X\rho X + \frac{p_0}{3} Y\rho Y + \frac{p_0}{3} Z\rho Z. \tag{5.4}$$

To compare the derived noise model to the Monte Carlo simulation I proceed as follows:

1. Initialise $\rho_{in} = |0\rangle \langle 0|$.
2. Determine the RF pulse $(\Omega, \phi, \Delta, t)$ that gives the rotation vector $\vec{R}$ associated with U_i, as in equation (4.32).
3. Using the procedure outlined in section 4.2.3, compute the Kraus operators

for U_i.

4. Evolve the state $\rho \mapsto \sum_n K_n U_i K_n^\dagger$.

5. Repeat steps 2-4 for each U_i to obtain the output state $\bar{\rho}_{out}$.

6. Calculate $\mathcal{F}_{TVD}$ between this output state and the output state from the Monte Carlo simulation.

To use the depolarising channel, we first need to determine an appropriate value for the parameter p_0. This is done using a small trial circuit. First, use a Monte Carlo simulation to determine the expected noisy output state ρ_{MC}. Then, analytically compute the output of the depolarising channel method in terms of p_0, $\rho_{DP}(p_0)$. Finally, maximise $\mathcal{F}_{TVD}(\rho_{MC}, \rho_{DP}(p_0)$ with respect to p_0. Once a suitable value of p_0 is found, the depolarising channel method then proceeds as follows:

1. Initialise $\rho_{in} = |0\rangle \langle 0|$.

2. Evolve the state $\rho \mapsto U_i \rho U_i^\dagger$.

3. Apply the depolarising channel $\rho \mapsto \mathcal{D}[\rho]$.

4. Repeat steps 2-3 for each U_i to obtain the output state $\bar{\rho}_{out}$.

5. Calculate $\mathcal{F}_{TVD}$ between this output state and the output state from the Monte Carlo simulation.

The circuits I use to compare the derived model to the depolarising channel comprise an alternating sequence of rotations about the x and y axes of the Bloch sphere with rotation angles selecting randomly from a uniform distribution $(0, \pi]$. I use our best guess for the size of the errors based on experimental observations, $\sigma_\Omega = \sigma_\phi = \sigma_\Delta = 10^{-3}$. In figure 5.1 I plot the fidelity of each noise model (compared to the Monte Carlo simulation) against number of gates in the circuit. As we can see, the derived model very accurately represents the output of

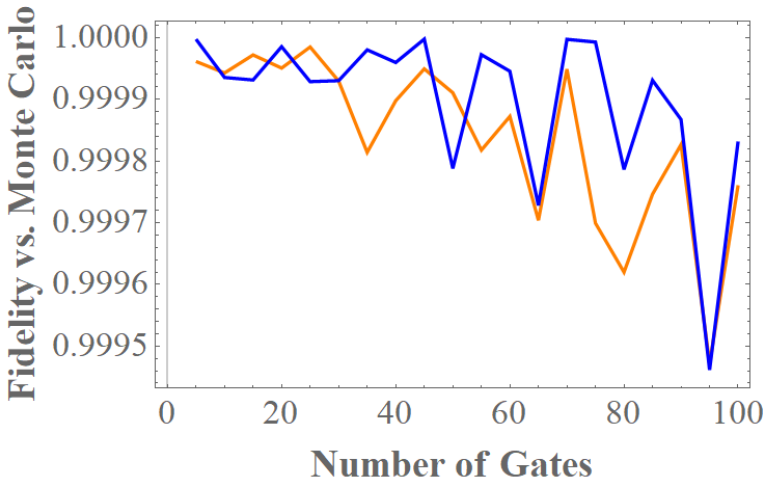


Figure 5.1: The fidelity of noisy quantum computer simulations is plotted against number of gates in a single-qubit circuit with $\sigma_\Omega = \sigma_\phi = \sigma_\Delta = 10^{-3}$. The fidelity of the derived model (orange) and the depolarising channel (blue) are calculated compared to a Monte Carlo simulation.

the noisy quantum computer as described by the Monte Carlo simulation with fidelities above 0.999 even for very large circuits. However, the derived model is outperformed by the very simple depolarising channel for most circuits simulated. One possible explanation for this is that just one parameter, p_0, is required to encompass the noisy behaviour of single-qubit gates. To investigate the relationship between p_0 and the underlying errors, I have plotted the fitted depolarising channel parameter p_0 as a function of the size of the errors σ for three different single-qubit circuits: 2 gates, 4 gates, and 8 gates. I am using σ as a shorthand for all three errors $(\sigma_\Omega, \sigma_\phi, \sigma_\Delta)$ having equal size. This is shown in figure 5.2. We observe that for small values of σ, the value of p_0 is very similar regardless of the single-qubit circuit before diverging slightly as σ increases. In the comparison of the derived model and the depolarising channel, I have used our best estimate for the size of the errors, $\sigma = 10^{-3}$. Figure 5.3 therefore suggest that the single parameter p_0 is sufficient to encompass the behaviour of noisy single-qubit circuits, since we are using a small value of σ. However, since there is no direct connection between p_0 and the underlying noise, this explanation still seems unlikely

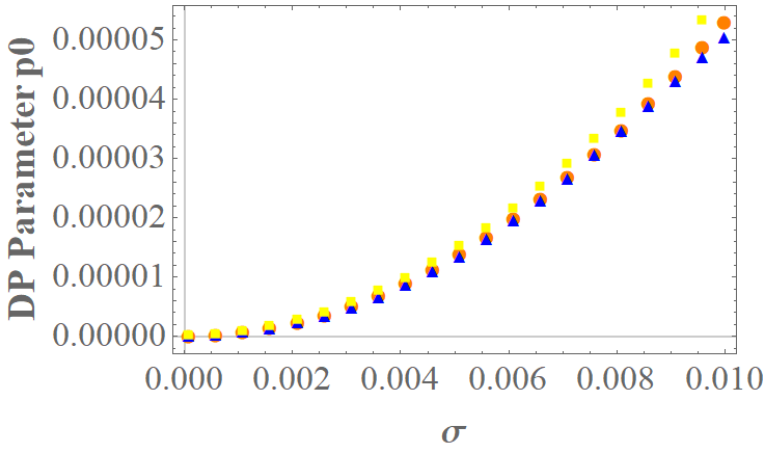


Figure 5.2: Fitted depolarising channel parameter p_0 plotted as a function of noise size σ for three different single-qubit gates: 2 gates (orange circles), 4 gates (blue triangles), 16 gates (yellow squares).

to be true and further work is required to determine if such a relationship exists implicitly. An alternative explanation for the depolarising channel outperforming the derived model for single-qubit circuits is that the errors introduced by $(\sigma_\Omega, \sigma_\phi, \sigma_\Delta)$ are small enough to mask the benefits of the derived model. By this, I mean that the errors are small enough so that the fidelity of the noisy circuit compared to the ideal circuit is already very close to 1. Then, because a very small value of p_0 provides a good approximation to the ideal circuit, it naturally provides a good approximation to the noisy circuit. One way to test this is to repeat the comparison with the size of the errors increased. This is shown in figure 5.3 for $\sigma_\Omega = \sigma_\phi = \sigma_\Delta = 0.1$. We again observe the depolarising channel outperforming the derived model for most circuit sizes. However, since we have increased the size of the errors, truncating the derived model at first order is almost certainly insufficient. Importantly, we see that with increased errors the depolarising channel performs significantly worse. Hence it is possible that the derived model can outperform the depolarising channel if it is derived to higher order.

The depolarising channel is a phenomenological model. However, I have shown that it does a good job at representing noise in single-qubit circuits when the

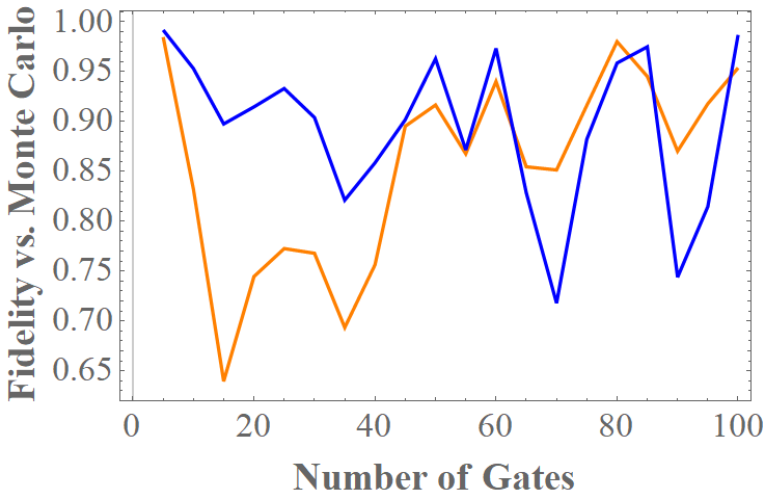


Figure 5.3: The fidelity of noisy quantum computer simulations is plotted against number of gates in a single-qubit circuit with $\sigma_\Omega = \sigma_\phi = \sigma_\Delta = 0.1$. The fidelity of the derived model (orange) and the depolarising channel (blue) are calculated compared to a Monte Carlo simulation.

errors are small. We predict that the benefits of the derived model will become apparent when it is extended to multi-qubit gates and circuits where phenomenological models such as the depolarising channel are known to perform poorly. Furthermore, by performing the derivation in chapter 4 beyond first order in $\delta\vec{R}$ should improve the performance of the derived model beyond the depolarising channel, even for single-qubit circuits. Both of these claims are potential directions for future work.

5.2 An Advantage of the Derived Model

Despite being outperformed by the depolarising channel, the derived model still offers its own advantages. In the derived model, there is a clear link between fidelity and the sources of noise in the quantum computer. This allows us to investigate how the each source of noise impacts the fidelity of the quantum computer. This can potentially guide future design choices in the quantum hardware to minimise the impact of noise and maximise fidelity. As an example,

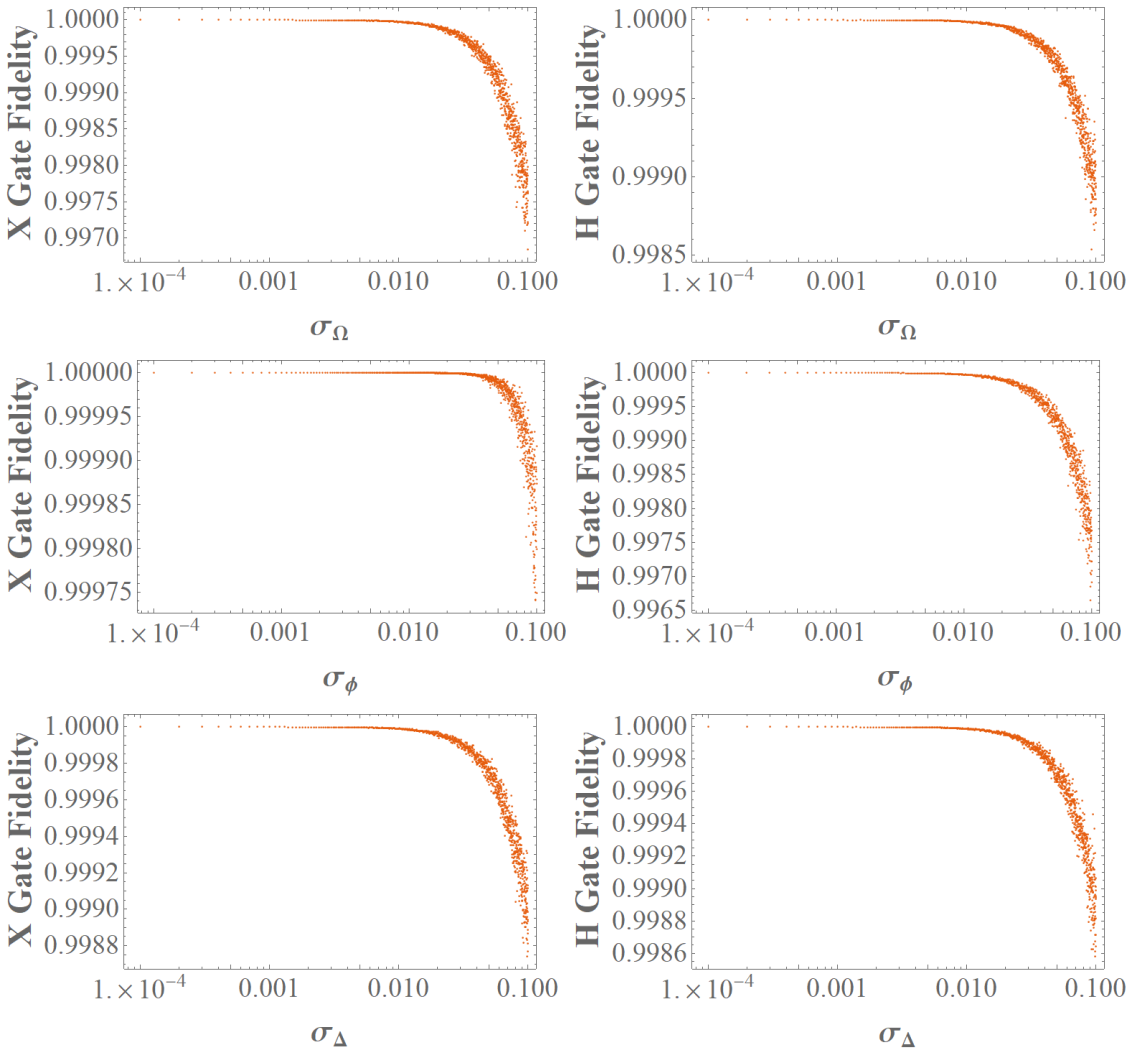


Figure 5.4: The fidelity of common gates, X and H, is plotted as a function of the magnitude of the control errors σ_Ω, σ_ϕ, and σ_Δ.

in figure 5.4 I have plotted the fidelity of two frequently used gates, X and H, as a function of the sources of noise σ_Ω, σ_ϕ, and σ_Δ. From this, we can see that the variance in the amplitude of the RF pulse σ_Ω has the greatest impact on the fidelity of the X gate. Conversely, the variance in the phase of the RF pulse σ_ϕ has the least impact for the X gate. In the case of the H gate, all sources of noise appear to be equally important to the fidelity. Our best estimate for each variance is $\sigma_\Omega = \sigma_\phi = \sigma_\Delta \approx 10^{-3}$. From figure 5.4, we observe that near 10^{-3}, the fidelity of the gates is linear with respect to the size of the noise,

and is very close to 1. This is reassuring because it means the errors will not compound quickly as we add additional gates to a quantum circuit. We are able to gain this information because the derived noise model has a direct connection to the underlying noise sources, unlike phenomenological models such as the depolarising channel.

In this chapter I have investigated the validity of the noise model derived in chapter 4. I have shown that for single-qubit circuits, the derived model is outperformed by a standard depolarising channel. However, I have also demonstrated the importance of having a noise model with a direct connection to the underlying sources of noise in the quantum computer. Future work should include re-deriving the noise model to higher orders in $\delta\vec{R}$ to see if it can outperform the depolarising channel even for single-qubit circuits. Furthermore, the derived model should be extended to multi-qubit gates, as this is where we predict the most advantage will be gained.

Chapter 6

Conclusions and Future Work

In this thesis, I have provided the first steps towards developing an accurate and efficient simulator for noisy diamond quantum computers with a direct connection to the underlying sources of noise. Such a simulator would be a powerful tool in near-term quantum computing research. Noisy simulations can be used the provide feedback on sources of noise to identify which have the greatest impact on fidelity. This information can then be used to guide future engineering choices to improve the performance of quantum hardware, and to optimise the compilation of quantum algorithms to minimise the effects of noise. Furthermore, classical simulations can be used to validate and benchmark the performance of quantum computers, both against classical computers and other quantum computers, and to develop new quantum algorithms. With this motivation in mind, this thesis provides two key contributions. Firstly, a comparison of current state of the art simulators that identified the most efficient. Secondly, a methodology for deriving a noise model for diamond quantum computers that incorporates the physics of the underlying sources of noise.

I achieved the first contribution in chapter 3. I compared two leading state of the art classical simulators (IBM's Aer and Oak Ridge National Laboratory's ExaTN-MPS) to identify which is most efficient for general-purpose quantum computing research. I defined a worst-case quantum circuit for MPS simulations,

a novel concept absent from the literature. I used this worst-case circuit to show that in some cases MPS simulators require computational resources exponential in the number of qubits, n, to provide high fidelity outputs. Then, by performing runtime comparisons for two useful quantum circuits, I showed that Aer significantly outperforms ExaTN-MPS for small n, at least $n \lesssim 30$. However, due to the theoretical scaling behaviour of the two simulators, I showed that ExaTN-MPS might overtake Aer within a range of qubits that we are interested in simulating, 50-100. This cross over came with a high degree of uncertainty due to the distance I had to extrapolate, and the discrepancies between the expected scaling behaviour and the observed scaling behaviour. Following these investigations, I concluded that while MPS simulations have been shown to be accurate and efficient for certain quantum systems, current Schrodinger simulators are faster at producing high fidelity outputs and should be the preferred choice for a general-purpose simulator.

There was some evidence that for larger n it may become valid to truncate the bond dimension of MPS simulations for the worst-case circuit. This would imply that the computational costs associated with MPS simulations may be made linear with n. Hence future work should use high-performance computing resources to run the worst-case circuit for larger n to verify whether my conclusion still holds, or if there exists is a fixed bond dimension which produces high fidelity simulations for all n. The runtime comparisons should also be performed with high-performance computing resources to attempt to identify the theorised cross over point without extrapolation. Another potential avenue for future work is investigating precisely which circuit characteristics govern whether MPS simulations are efficient. These extensions of my work could lead to a simulator that employs a Schrodinger method or a MPS method depending on the input circuit.

I achieved the second contribution in chapters 4 and 5. In chapter 4, I derived a

Kraus model for noisy diamond quantum computers that has a direct connection to the underlying sources of noise. I first showed how to convert from a master equation description of system-environment interactions to a Kraus model. Then I demonstrated a novel derivation method to construct a Kraus model for single-qubit control errors resulting from noise in the RF pulses that effect single-qubit gates. I was able to obtain an analytic form of this model for the universal set of gates used in diamond quantum computing. This method is generalisable, so it can be used for other sources of noise, as well as for other quantum computer architectures. In chapter 5, by simulating single-qubit circuits, I showed that while the derived model provides high fidelity simulations of noisy quantum computers, it is outperformed by the simple depolarising channel. This result was attributed to the small error sizes in single-qubit circuits favouring the fitted depolarising channel, which would not be true for larger errors including those associated with multi-qubit circuits. Despite this result, the derived model still offers advantages over phenomenological models due to its direct connection with the underlying noise sources. I demonstrated this by finding the fidelity of some single-qubit gates as a function of noise.

Future work should extend my derived model to higher-order terms to improve its fidelity of simulating single-qubit circuits. Then, the noise model for diamond quantum computers should be completed by incorporating multi-qubit control errors. The completed derived model should then be compared with existing, phenomenological noisy simulators to validate that it more accurately reflects the output of noisy, multi-qubit circuits. If the completed noise model is shown to be more accurate than existing simulators as predicted, it should be incorporated into an efficient, state of the art simulator such as Aer. This accurate, efficient noisy simulator could then be used to more deeply investigate gate fidelities as a function of noise with the goal of improving quantum hardware. Moreover, it could be used for a wide range of near-term quantum computing research as outlined above.

Appendix A

Proof of Proposition 1

To prove Proposition 1 we need the following lemma,

Lemma A.1: *For any vector* $\vec{R}$*,*

$$(\vec{R}\cdot\vec{\sigma})^2 = |\vec{R}|^2, \tag{A.1}$$

where $\vec{\sigma}$ is the vector of Pauli matrices.

Proof. Let $\vec{R} = (R_x, R_y, R_z)$ so that

$$\begin{aligned}
(\vec{R}\cdot\vec{\sigma})^2 &= (R_xX + R_yY + R_zZ)^2 && \text{(A.2)}\\
&= R_x^2X^2 + R_xR_yXY + R_xR_zXZ + R_yR_xYX + R_y^2Y^2 && \text{(A.3)}\\
&\qquad + R_yR_zYZ + R_zR_xZX + R_zR_yZY + R_z^2Z^2 && \text{(A.4)}\\
&= R_x^2 + R_y^2 + R_z^2 && \text{(A.5)}\\
&= |\vec{R}|^2 && \text{(A.6)}
\end{aligned}$$

where in the third line we've used the Pauli anti-commutation relations $\{\sigma_i, \sigma_j\} = 2\delta_{ij}I$. □

Proposition 1: *With everything as defined in the text,*

$$(\mathcal{L}_{-\frac{i}{2}\vec{R}\cdot\vec{\sigma}})^k\left(-\frac{i}{2}\delta\vec{R}\cdot\vec{\sigma}\right) = \begin{cases} (-\frac{1}{2}\gamma^2)^{\frac{k-1}{2}}\mathcal{L}_{-\frac{i}{2}\vec{R}\cdot\vec{\sigma}}\left(-\frac{i}{2}\delta\vec{R}\cdot\vec{\sigma}\right) & k\geq 1 \text{ is odd} \\ (-\frac{1}{2}\gamma^2)^{\frac{k}{2}-1}(\mathcal{L}_{-\frac{i}{2}\vec{R}\cdot\vec{\sigma}})^2\left(-\frac{i}{2}\delta\vec{R}\cdot\vec{\sigma}\right) & k\geq 2 \text{ is even} \end{cases}.$$

Proof. This will be a proof by induction on k. The first two base cases $k = 1$ and $k = 2$ are trivially true. It will be useful to expand these commutators,

$$(\mathcal{L}_{-\frac{i}{2}\vec{R}\cdot\vec{\sigma}})^1\left(-\frac{i}{2}\delta\vec{R}\cdot\vec{\sigma}\right) = \left[-\frac{i}{2}\vec{R}\cdot\vec{\sigma}, -\frac{i}{2}\delta\vec{R}\cdot\vec{\sigma}i\right] \tag{A.7}$$

$$= -\frac{1}{4}\left((\vec{R}\cdot\vec{\sigma})(\delta\vec{R}\cdot\vec{\sigma}) - (\delta\vec{R}\cdot\vec{\sigma})(\vec{R}\cdot\vec{\sigma})\right), \tag{A.8}$$

and

$$(\mathcal{L}_{-\frac{i}{2}\vec{R}\cdot\vec{\sigma}})^2\left(-\frac{i}{2}\delta\vec{R}\cdot\vec{\sigma}\right) = \left[-\frac{i}{2}\vec{R}\cdot\vec{\sigma}, \left[-\frac{i}{2}\vec{R}\cdot\vec{\sigma}, -\frac{i}{2}\delta\vec{R}\cdot\vec{\sigma}\right]\right] \tag{A.9}$$

$$= \frac{i}{4}\left(\gamma^2\delta\vec{R}\cdot\vec{\sigma} - (\vec{R}\cdot\vec{\sigma})(\delta\vec{R}\cdot\vec{\sigma})(\vec{R}\cdot\vec{\sigma})\right), \tag{A.10}$$

where we have used Lemma A.1 and $\gamma \equiv |\vec{R}|$ in the last line. For the induction step, we will also need to directly prove the proposition for $k = 3$ and $k = 4$. First,

$$(\mathcal{L}_{-\frac{i}{2}\vec{R}\cdot\vec{\sigma}})^3\left(-\frac{i}{2}\delta\vec{R}\cdot\vec{\sigma}\right) = \left[-\frac{i}{2}\vec{R}\cdot\vec{\sigma}, (\mathcal{L}_{-\frac{i}{2}\vec{R}\cdot\vec{\sigma}})^2\left(-\frac{i}{2}\delta\vec{R}\cdot\vec{\sigma}\right)\right] \tag{A.11}$$

$$= -\frac{1}{2}\gamma^2\left(\frac{1}{4}(\delta\vec{R}\cdot\vec{\sigma})(\vec{R}\cdot\vec{\sigma}) - \frac{1}{4}(\vec{R}\cdot\vec{\sigma})(\delta\vec{R}\cdot\vec{\sigma})\right) \tag{A.12}$$

$$= -\frac{1}{2}\gamma^2(\mathcal{L}_{-\frac{i}{2}\vec{R}\cdot\vec{\sigma}})^1\left(-\frac{i}{2}\delta\vec{R}\cdot\vec{\sigma}\right), \tag{A.13}$$

verifying the proposition for $k = 3$. Next,

$$(\mathcal{L}_{-\frac{i}{2}\vec{R}\cdot\vec{\sigma}})^4\left(-\frac{i}{2}\delta\vec{R}\cdot\vec{\sigma}\right) = \left[-\frac{i}{2}\vec{R}\cdot\vec{\sigma}, (\mathcal{L}_{-\frac{i}{2}\vec{R}\cdot\vec{\sigma}})^3\left(-\frac{i}{2}\delta\vec{R}\cdot\vec{\sigma}\right)\right] \tag{A.14}$$

$$= \left[-\frac{i}{2}\vec{R}\cdot\vec{\sigma}, -\gamma^2(\mathcal{L}_{-\frac{i}{2}\vec{R}\cdot\vec{\sigma}})^1\left(-\frac{i}{2}\delta\vec{R}\cdot\vec{\sigma}\right)\right] \tag{A.15}$$

$$= -\frac{1}{2}\gamma^2\left(\frac{i}{4}\gamma^2(\delta\vec{R}\cdot\vec{\sigma}) - \frac{i}{4}(\vec{R}\cdot\vec{\sigma})(\delta\vec{R}\cdot\vec{\sigma})(\vec{R}\cdot\vec{\sigma})\right) \tag{A.16}$$

$$= -\frac{1}{2}\gamma^2(\mathcal{L}_{-\frac{i}{2}\vec{R}\cdot\vec{\sigma}})^2\left(-\frac{i}{2}\delta\vec{R}\cdot\vec{\sigma}\right), \tag{A.17}$$

verifying the proposition for $k = 4$. Now we can move on to the induction step. We will treat the odd and even cases separately. Let $k > 3$ be odd. We assume

the proposition holds for $k-2$ and want to prove it is true for k. We have

$$(\mathcal{L}_{-\frac{i}{2}\vec{R}\cdot\vec{\sigma}})^k\left(-\frac{i}{2}\delta\vec{R}\cdot\vec{\sigma}\right) = \left[-\frac{i}{2}\vec{R}\cdot\vec{\sigma},\left[-\frac{i}{2}\vec{R}\cdot\vec{\sigma},(\mathcal{L}_{-\frac{i}{2}\vec{R}\cdot\vec{\sigma}})^{k-2}\left(-\frac{i}{2}\delta\vec{R}\cdot\vec{\sigma}\right)\right]\right] \tag{A.18}$$

$$= \left[-\frac{i}{2}\vec{R}\cdot\vec{\sigma},\left[-\frac{i}{2}\vec{R}\cdot\vec{\sigma},(-\gamma^2)^{\frac{k-3}{2}}(\mathcal{L}_{-\frac{i}{2}\vec{R}\cdot\vec{\sigma}})^1\left(-\frac{i}{2}\delta\vec{R}\cdot\vec{\sigma}\right)\right]\right] \tag{A.19}$$

$$= (-\frac{1}{2}\gamma^2)^{\frac{k-3}{2}}(\mathcal{L}_{-\frac{i}{2}\vec{R}\cdot\vec{\sigma}})^3\left(-\frac{i}{2}\delta\vec{R}\cdot\vec{\sigma}\right) \tag{A.20}$$

$$= (-\frac{1}{2}\gamma^2)^{\frac{k-1}{2}}(\mathcal{L}_{-\frac{i}{2}\vec{R}\cdot\vec{\sigma}})^1\left(-\frac{i}{2}\delta\vec{R}\cdot\vec{\sigma}\right) \tag{A.21}$$

as desired. Next let $k>4$ be even. We assume the proposition holds for $k-2$ and want to prove it is true for k. We have

$$(\mathcal{L}_{-\frac{i}{2}\vec{R}\cdot\vec{\sigma}})^k\left(-\frac{i}{2}\delta\vec{R}\cdot\vec{\sigma}\right) = \left[-\frac{i}{2}\vec{R}\cdot\vec{\sigma},\left[-\frac{i}{2}\vec{R}\cdot\vec{\sigma},(\mathcal{L}_{-\frac{i}{2}\vec{R}\cdot\vec{\sigma}})^{k-2}\left(-\frac{i}{2}\delta\vec{R}\cdot\vec{\sigma}\right)\right]\right] \tag{A.22}$$

$$= \left[-\frac{i}{2}\vec{R}\cdot\vec{\sigma},\left[-\frac{i}{2}\vec{R}\cdot\vec{\sigma},(-\gamma^2)^{\frac{k}{2}-2}(\mathcal{L}_{-\frac{i}{2}\vec{R}\cdot\vec{\sigma}})^2\left(-\frac{i}{2}\delta\vec{R}\cdot\vec{\sigma}\right)\right]\right] \tag{A.23}$$

$$= (-\frac{1}{2}\gamma^2)^{\frac{k}{2}-2}(\mathcal{L}_{-\frac{i}{2}\vec{R}\cdot\vec{\sigma}})^4\left(-\frac{i}{2}\delta\vec{R}\cdot\vec{\sigma}\right) \tag{A.24}$$

$$= (-\frac{1}{2}\gamma^2)^{\frac{k}{2}-1}(\mathcal{L}_{-\frac{i}{2}\vec{R}\cdot\vec{\sigma}})^2\left(-\frac{i}{2}\delta\vec{R}\cdot\vec{\sigma}\right) \tag{A.25}$$

as desired. □

Appendix B

Derived Kraus Operators for Single-Qubit Gates

In chapter 4, we considered an arbitrary single-qubit gate U' as a rotation of the Bloch sphere,

$$U' = e^{-\frac{i}{2}(\vec{R}+\delta\vec{R})\cdot\vec{\sigma}}, \tag{B.1}$$

where $\vec{R}$ is the ideal axis of rotation, $\delta\vec{R}$ is a small error due to control errors, and $\vec{\sigma}$ is the vector of Pauli matrices. We can write $\vec{R}$ and $\delta\vec{R}$ in terms of the parameters of the control pules, Ω, ϕ, Δ, t as in the main text. Recall that the rotation angle is

$$\gamma = |\vec{R}| = \sqrt{\Omega^2 + \Delta^2}t, \tag{B.2}$$

and we also defined F and G as the hypergeometric functions,

$$F(\gamma) = \sum_{n \text{ odd}} \frac{(-1)^n}{(n+1)!}(-\frac{1}{2}\gamma^2)^{n-1}, \tag{B.3}$$

$$G(\gamma) = \sum_{n \text{ even}} \frac{(-1)^n}{(n+1)!}(-\frac{1}{2}\gamma^2)^{n-1}. \tag{B.4}$$

Finally, for convenience we also defined

$$K_1 = \sigma_\Omega t \cos\phi X + \sigma_\Omega t \sin\phi Y, \tag{B.5}$$

$$K_2 = \sigma_\phi \Omega t \sin\phi X - \sigma_\phi \Omega t \cos\phi Y, \tag{B.6}$$

$$K_3 = \sigma_\Delta t Z. \tag{B.7}$$

Then, following the derivation in chapter 4.2, the Kraus operators for the noisy rotation are given by:

$$E_1 = I \tag{B.8}$$

$$E_2 = \sqrt{c_1} K_1 \tag{B.9}$$

$$E_3 = \sqrt{c_1} K_2 \tag{B.10}$$

$$E_4 = \sqrt{c_1} K_3 \tag{B.11}$$

$$E_5 = \sqrt{c_2} (\vec{R} \cdot \vec{\sigma}) K_1 \tag{B.12}$$

$$E_6 = \sqrt{c_2} (\vec{R} \cdot \vec{\sigma}) K_2 \tag{B.13}$$

$$E_7 = \sqrt{c_2} (\vec{R} \cdot \vec{\sigma}) K_3 \tag{B.14}$$

$$E_8 = \sqrt{c_2} K_1 (\vec{R} \cdot \vec{\sigma}) \tag{B.15}$$

$$E_9 = \sqrt{c_2} K_2 (\vec{R} \cdot \vec{\sigma}) \tag{B.16}$$

$$E_{10} = \sqrt{c_2} K_3 (\vec{R} \cdot \vec{\sigma}) \tag{B.17}$$

$$E_{11} = \sqrt{c_3} \Big((\vec{R} \cdot \vec{\sigma}) K_1 (\vec{R} \cdot \vec{\sigma}) \Big) \tag{B.18}$$

$$E_{12} = \sqrt{c_3} \Big((\vec{R} \cdot \vec{\sigma}) K_2 (\vec{R} \cdot \vec{\sigma}) \Big) \tag{B.19}$$

$$E_{13} = \sqrt{c_3} \Big((\vec{R} \cdot \vec{\sigma}) K_3 (\vec{R} \cdot \vec{\sigma}) \Big) \tag{B.20}$$

$$E_{14} = \sqrt{c_4} \Big(K_1 (\vec{R} \cdot \vec{\sigma}) - i (\vec{R} \cdot \vec{\sigma}) K_1 (\vec{R} \cdot \vec{\sigma}) \Big) \tag{B.21}$$

$$E_{15} = \sqrt{c_4} \Big(K_2 (\vec{R} \cdot \vec{\sigma}) - i (\vec{R} \cdot \vec{\sigma}) K_2 (\vec{R} \cdot \vec{\sigma}) \Big) \tag{B.22}$$

$$E_{16} = \sqrt{c_4} \Big(K_3 (\vec{R} \cdot \vec{\sigma}) - i (\vec{R} \cdot \vec{\sigma}) K_3 (\vec{R} \cdot \vec{\sigma}) \Big) \tag{B.23}$$

$$E_{17} = \sqrt{c_5} \Big((\vec{R} \cdot \vec{\sigma}) K_1 - K_1 (\vec{R} \cdot \vec{\sigma}) \Big) \tag{B.24}$$

$$E_{18} = \sqrt{c_5} \Big((\vec{R} \cdot \vec{\sigma}) K_2 - K_2 (\vec{R} \cdot \vec{\sigma}) \Big) \tag{B.25}$$

$$E_{19} = \sqrt{c_5} \Big((\vec{R} \cdot \vec{\sigma}) K_3 - K_3 (\vec{R} \cdot \vec{\sigma}) \Big) \tag{B.26}$$

$$E_{20} = \sqrt{c_5} \Big(i (\vec{R} \cdot \vec{\sigma}) K_1 - i K_1 (\vec{R} \cdot \vec{\sigma}) + K_1 \Big) \tag{B.27}$$

$$E_{21} = \sqrt{c_5}\left(i(\vec{R}\cdot\vec{\sigma})K_2 - iK_2(\vec{R}\cdot\vec{\sigma}) + K_2\right) \tag{B.28}$$

$$E_{22} = \sqrt{c_5}\left(i(\vec{R}\cdot\vec{\sigma})K_3 - iK_2(\vec{R}\cdot\vec{\sigma}) + K_3\right) \tag{B.29}$$

$$E_{23} = \sqrt{c_6}(K_1 + \vec{R}\cdot\vec{\sigma})K_1(\vec{R}\cdot\vec{\sigma}) \tag{B.30}$$

$$E_{24} = \sqrt{c_6}(K_2 + \vec{R}\cdot\vec{\sigma})K_2(\vec{R}\cdot\vec{\sigma}) \tag{B.31}$$

$$E_{25} = \sqrt{c_6}(K_3\vec{R}\cdot\vec{\sigma})K_3(\vec{R}\cdot\vec{\sigma}) \tag{B.32}$$

$$E_{26} = \sqrt{c_7}\left(K_1(\vec{R}\cdot\vec{\sigma}) + (\vec{R}\cdot\vec{\sigma})K_1\right) \tag{B.33}$$

$$E_{27} = \sqrt{c_7}\left(K_2(\vec{R}\cdot\vec{\sigma}) + (\vec{R}\cdot\vec{\sigma})K_2\right) \tag{B.34}$$

$$E_{28} = \sqrt{c_7}\left(K_3(\vec{R}\cdot\vec{\sigma}) + (\vec{R}\cdot\vec{\sigma})K_3\right) \tag{B.35}$$

$$E_{29} = \sqrt{c_4}\left(i(\vec{R}\cdot\vec{\sigma})K_1(\vec{R}\cdot\vec{\sigma}) + (\vec{R}\cdot\vec{\sigma})K_1\right) \tag{B.36}$$

$$E_{30} = \sqrt{c_4}\left(i(\vec{R}\cdot\vec{\sigma})K_2(\vec{R}\cdot\vec{\sigma}) + (\vec{R}\cdot\vec{\sigma})K_2\right) \tag{B.37}$$

$$E_{31} = \sqrt{c_4}\left(i(\vec{R}\cdot\vec{\sigma})K_3(\vec{R}\cdot\vec{\sigma}) + (\vec{R}\cdot\vec{\sigma})K_3\right) \tag{B.38}$$

where the constants c_i are

$$c_1 = \frac{1}{16}(G(\gamma)\gamma^2 - 2)(1 + G(\gamma) - F(\gamma)) \tag{B.39}$$

$$c_2 = \frac{1}{16}(2F(\gamma)^2 - F(\gamma)G(\gamma)) \tag{B.40}$$

$$c_3 = \frac{1}{16}(G(\gamma)^2 - 2F(\gamma)G(\gamma) + G(\gamma)(G(\gamma)\gamma^2 - 2) \tag{B.41}$$

$$c_4 = \frac{1}{16}F(\gamma)G(\gamma) \tag{B.42}$$

$$c_5 = \frac{1}{16}F(\gamma)(G(\gamma)\gamma^2 - 2) \tag{B.43}$$

$$c_6 = \frac{1}{16}G(\gamma)(2 - G(\gamma)\gamma^2) \tag{B.44}$$

$$c_7 = -\frac{1}{16}F(\gamma)^2 \tag{B.45}$$

By following the procedure outlined at the end of chapter 4.2, these operators can be numerically converted into an equivalent set of 4 Kraus operators for use in noisy simulations.

Bibliography

[1] National Academy of Engineering. *Frontiers of Engineering: Reports on Leading-Edge Engineering from the 2018 Symposium*. National Academies Press, Washington, D.C., January 2019.

[2] A S Holevo. Bounds for the Quantity of Information Transmitted by a Quantum Communication Channel. *Probl. Peredachi Inf.*, 9(3):3–11, 1973.

[3] R. S. Ingarden. Quantum Information Theory. *Reports on Mathematical Physics*, 10(1):43–72, August 1976.

[4] P. Benioff. The computer as a physical system: A microscopic quantum mechanical Hamiltonian model of computers as represented by Turing machines. *Journal of Statistical Physics*, 22(5):563–591, May 1980.

[5] R. P. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21:467–488, June 1982.

[6] P.W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 124–134, November 1994.

[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*, pages 212–219, Philadelphia, Pennsylvania, United States, 1996. ACM Press.

[8] R.D. Rosenwald, D.A. Meyer, and H.A. Schmitt. Applications of quantum algorithms to partially observable Markov decision processes. In *2004 5th*

Asian Control Conference (IEEE Cat. No.04EX904), volume 1, pages 420–427 Vol.1, July 2004.

[9] R. Orus, S. Mugel, and E. Lizaso. Quantum computing for finance: Overview and prospects. *Reviews in Physics*, 4, November 2019.

[10] D. Solenov, J. Brieler, and J. F. Scherrer. The Potential of Quantum Computing and Machine Learning to Advance Clinical Research and Change the Practice of Medicine. *Missouri Medicine*, 115(5):463–467, 2018.

[11] I. Kassal, S. P. Jordan, P. J. Love, and *et al.* Polynomial-time quantum algorithm for the simulation of chemical dynamics. *Proceedings of the National Academy of Sciences*, 105(48):18681–18686, December 2008.

[12] P. Krantz, M. Kjaergaard, F. Yan, and *et al.* A Quantum Engineer's Guide to Superconducting Qubits. *Applied Physics Reviews*, 6(2):021318, June 2019.

[13] C. D. Bruzewicz, J Chiaverini, R. McConnell, and *et al.* Trapped-Ion Quantum Computing: Progress and Challenges. *Applied Physics Reviews*, 6(2): 021314, June 2019.

[14] S. Slussarenko and G. J. Pryde. Photonic quantum information processing: A concise review. *Applied Physics Reviews*, 6(4):041303, December 2019.

[15] S. Pezzagna and J. Meijer. Quantum computer based on color centers in diamond. *Applied Physics Reviews*, 8(1):011308, March 2021.

[16] T. H. Taminiau, J. Cramer, T. van der Sar, and *et. al.* Universal control and error correction in multi-qubit spin registers in diamond. *Nature Nanotechnology*, 9(3):171–176, March 2014.

[17] K. Xu, T. Xie, Z. Li, and *et. al.* Experimental Adiabatic Quantum Factorization under Ambient Conditions Based on a Solid-State Single Spin System. *Physical Review Letters*, 118(13):130504, March 2017.

[18] Y. Wang, F. Dolde, J. Biamonte, and *et. al.* Quantum Simulation of Helium Hydride Cation in a Solid-State Spin Register. *ACS Nano*, 9(8):7769–7774, August 2015.

[19] J. Preskill. Quantum Computing in the NISQ era and beyond. *Quantum*, 2:79, August 2018.

[20] F. Arute, K. Arya, R. Babbush, and *et al.* Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510, October 2019.

[21] I. L. Markov, A. Fatima, S. V. Isakov, and *et al.* Quantum Supremacy Is Both Closer and Farther than It Appears. *arXiv:1807.10749 [quant-ph]*, September 2018.

[22] I. L. Chuang, N. A. Gershenfeld, and M. Kubinec. Experimental implementation of fast quantum searching. *Physical Review Letters*, 80:3408–3411, 1998.

[23] M. N. Lilly and T. S. Humble. Modeling Noisy Quantum Circuits Using Experimental Characterization. *arXiv:2001.08653 [quant-ph]*, January 2020.

[24] K. Georgopoulos, C. Emary, and P. Zuliani. Modelling and Simulating the Noisy Behaviour of Near-term Quantum Computers. *arXiv:2101.02109 [quant-ph]*, February 2021.

[25] M. A. Nielsen and I. L. Chuang. *Quantum computation and quantum information*. Cambridge University Press, Cambridge ; New York, 10th anniversary ed edition, 2010.

[26] C. M. Dawson and M. A. Nielsen. The Solovay-Kitaev algorithm. *arXiv:quant-ph/0505030*, August 2005.

[27] Y. Chen, S. Stearn, S. Vella, and *et. al.* Optimisation of diamond quantum processors. *New Journal of Physics*, 22(9):093068, September 2020.

[28] P. Neumann, J. Beck, and M. Seiner. Single-shot readout of a single nuclear spin. *Science*, 329:542–544, July 2010.

[29] D. A. Lidar. Lecture Notes on the Theory of Open Quantum Systems. *arXiv:1902.00967 [quant-ph]*, February 2020.

[30] T. F. Havel. Robust procedures for converting among Lindblad, Kraus and matrix representations of quantum dynamical semigroups. *J. Math. Phys.*, 44(2):25, 2003.

[31] Y. I. Bogdanov, A. Y. Chernyavskiy, A. S. Holevo, and *et. al.* Mathematical modeling of quantum noise and the quality of hardware compo- nents of quantum computers. *International Conference Micro- and Nano-Electronics 2012*, January 2013.

[32] M. Jiang, S. Luo, and S. Fu. Channel-state duality. *Physical Review A*, 87 (2):022310, February 2013.

[33] S. Aaronson and D. Gottesman. Improved simulation of stabilizer circuits. *Physical Review A*, 70(5):052328, November 2004.

[34] B. Villalonga, D. Lyakh, S. Boixo, and *et. al.* Establishing the Quantum Supremacy Frontier with a 281 Pflop/s Simulation. *Quantum Science and Technology*, 5(3):034003, April 2020.

[35] Z. Chen, Q. Zhou, C. Xue, and *et. al.* 64-qubit quantum circuit simulation. *Science Bulletin*, 63(15):964–971, August 2018.

[36] J. Chen, F. Zhang, C. Huang, and *et. al.* Classical Simulation of Intermediate-Size Quantum Circuits. *arXiv:1805.01450 [quant-ph]*, May 2018.

[37] U. Schollwoeck. The density-matrix renormalization group in the age of matrix product states. *Annals of Physics*, 326(1):96–192, January 2011.

[38] I. L. Markov and Y. Shi. Simulating Quantum Computation by Contracting Tensor Networks. *SIAM Journal on Computing*, 38(3):963–981, January 2008.

[39] A. Dang, C. D. Hill, and L. C. L. Hollenberg. Optimising Matrix Product State Simulations of Shor's Algorithm. *Quantum*, 3:116, January 2019.

[40] A. McCaskey, E. Dumitrescu, M. Chen, and *et. al.* Validating quantum-classical programming models with tensor network simulations. *PLOS ONE*, 13(12):e0206704, December 2018.

[41] S. Cheng, C. Cao, C. Zhang, and *et. al.* Simulating Noisy Quantum Circuits with Matrix Product Density Operators. *arXiv:2004.02388 [cond-mat, physics:quant-ph]*, November 2020.

[42] R. Orus. A Practical Introduction to Tensor Networks: Matrix Product States and Projected Entangled Pair States. *Annals of Physics*, 349:117–158, October 2014.

[43] J. Biamonte. Lectures on Quantum Tensor Networks. *arXiv:1912.10049 [cond-mat, physics:math-ph, physics:quant-ph]*, January 2020.

[44] T. Roughgarden and G. Valiant. CS168: The Modern Algorithmic Toolbox Lecture #9: The Singular Value Decomposition (SVD) and Low-Rank Matrix Approximations. page 11, 2021.

[45] S. Ran, E. Tirrito, C. Peng, and *et. al.* Lecture Notes of Tensor Network Contractions. *arXiv:1708.09213 [cond-mat, physics:physics, physics:quant-ph]*, 964, 2020.

[46] F. Verstraete, J. J. GarcÃŋa-Ripoll, and J. I. Cirac. Matrix Product Density Operators: Simulation of Finite-Temperature and Dissipative Systems. *Physical Review Letters*, 93(20):207204, November 2004.

[47] S. N. Saadatmand, S. Yin, and M. L. Walker. qbOS: a Python framework for the development of coprocessing quantum-classical algorithms. In *First International Workshop on Integrating High-Performance and Quantum Computing (WIHPQC21), IEEE Quantum Week*, October 2021.

[48] J. Biamonte and V. Bergholm. Tensor Networks in a Nutshell. *arXiv:1708.00006 [cond-mat, physics:gr-qc, physics:hep-th, physics:math-ph, physics:quant-ph]*, July 2017.

[49] M. Saeedi and M. Pedram. Linear-depth quantum circuits for n -qubit Toffoli gates with no ancilla. *Physical Review A*, 87(6):062318, June 2013.

[50] S. Paeckel, T. KÃűhler, A. Swoboda, and *et. al.* Time-evolution methods for matrix-product states. *Annals of Physics*, 411:167998, December 2019.

[51] M. Ben-Or and A. Hassidim. Fast quantum byzantine agreement. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing - STOC '05*, page 481, Baltimore, MD, USA, 2005. ACM Press.

[52] S. Filipp. Qubit manipulation and rotating wave approximation. `https://qudev.phys.ethz.ch/static/content/courses/QSIT12/QSIT12_qubitmanipulation.pdf`, 2012.

[53] T. Kimura. Explicit Description of the Zassenhaus Formula. *Progress of Theoretical and Experimental Physics*, 2017(4), April 2017.