Department of Physics

Theoretical Physics Msci

Final Year Project

May 22, 2020

Single Qubit Randomised Benchmarking of Non-Clifford Gates

Author: Darren D Q Ng

Supervisor: Dr Peter Turner

Word Count: 9152

For submission in 2020

Contents

[1] Heavily adapted from interim report "Single Qubit Randomised Benchmarking of Non-Clifford Gates" [1].

ACKNOWLEDGEMENTS

First and foremost, I would like to thank Dr Peter Turner immensely for his supervision throughout the course of the project. His help, guidance and feedback were invaluable to my project. Secondly, I would like to thank Dr Martin Gradhand for his useful constructive criticisms on my interim report. I would like to also thank Dr Robin Harper from University of Sydney for the discussion and recommendation on the type of noise channel that was used in my randomised benchmarking simulation. Finally, I feel incredibly grateful towards the physics department for their understanding and allowing for any appropriate adjustments in these trying times, affected by the COVID-19 pandemic.

DECLARATIONS AND MITIGATIONS

All work contained within this project was primarily produced by myself, unless explicitly specified otherwise. Regular guidance and direction were provided by Dr Peter Turner during our weekly one hour meeting, where the progress and future direction of the project were subsequently discussed, but otherwise all the results were fundamentally my own creation. The simulation code was all written from scratch by myself in Python, with the help of existing libraries such as NumPy, SciPy and Matplotlib. In addition, due to the resource intensiveness of the computation, the module MPI4PY was adapted so it can be ran on the university's supercomputer BlueCrystalp3. The code used for generating data and plots as well as data analysis is presented in the appendix. The specific methodology of noise map generation in my simulation was recommended by Dr Robin Harper of University of Sydney.

In this report, some parts of the introduction and theory were adapted from my interim report [1]. For pages where it was more heavily adapted, a footnote was used to indicate acknowledgement with a citation included. In any case, I declare that this report is composed solely by myself and credits have been given where it is due.

In light of the recent UCU industrial strike action and the current COVID-19 pandemic, the project has been affected to some extent. Two weeks of meeting were cancelled under no notice due to the strike in 2019 and the university's premises were shut due to the pandemic in March 2020. The project was mathematical and highly computational in nature. Thus, the significant reduction of meeting time in person has hindered my opportunity to have productive discussions on verifying and asking questions about the preliminary results. The access to an appropriate working environment such as the computer room was also suddenly affected, which became difficult to deal with.

ABSTRACT

To build a universal quantum computer, it is critical to characterise and optimise the error rates of quantum gates. The technique of randomised benchmarking (RB) protocol is now the *de facto* standard, which implements using quantum gates from the Clifford group which satisfy a mathematical requirement of being a unitary 2-design. This allows for the protocol to be efficient, scalable and yet robust against state preparation and measurement errors. However, non-Clifford gates are required for attaining universal quantum computation in addition. In this project, a recently conjectured t-designs which forms a hierarchy of non-Clifford gate sets are investigated and analysed by the RB protocol. This project showed that these specific non-Clifford gate sets can indeed be efficiently characterised by the RB protocol, estimating their average gate fidelity to a high degree of accuracy.

Chapter 1

Introduction

1.1 Motivation[1]

The idea of a universal quantum computer was initially conceptualised by Feynman [2] in 1981. Since then, the potential of its physical realisation has garnered a lot of interest due to its prospective ability to significantly outperform classical computers, on tasks such as unstructured searching by Grover's algorithm, or factoring integers using the prominent Shor's algorithm [3–5]. Due to interactions with the inherent presence of random noise in any external environment, it is not possible for a quantum computer to be a perfectly closed system. Therefore the evolution of its system can no longer be described under the Schrödinger equation, since the system is no longer isolated and non-interacting. Rather, it needs a different mathematical framework described by a *quantum channel*. Some of the undesirable features of experimental outcomes such as decoherence of the system due to external noise, coupled with flawed implementations of quantum gates, together pose a potential threat to the construction of a universal quantum computer [6]. Despite a lot of efforts made in combating these limitations such as developing quantum error correction codes [7–10], they remain an active field of research with the goal of ultimately achieving a *fault-tolerant* quantum computer [11–13].

Quantum information is processed under quantum channels, therefore the control of errors during a computation is paramount in improving the efficiency of a quantum computer. In order to improve a quantum channel's quality, its best possible level of error control must first be characterised. Although the complete characterisation of errors in a quantum system is possible by traditional methods such as quantum process tomography (QPT) [14] and gate set tomography (GST) [15], with their demonstrations over different candidates of small number of qubits already achieved [16–20]; these implementations nevertheless present several difficulties. Firstly, these methods cannot differentiate the errors induced by state preparation and measurement (SPAM), and thus cannot ascertain if the resulted error analysis is solely attributed to the quantum channels themselves. Moreover, they often require precise knowledge of the system's SPAM, the lack of which often led to either systematic errors in operations or non-linear estimation issues, both of which are hard to resolve [21–23]. Secondly, they lack experimental scalability. They become practically infeasible as the number of qubits increases, making their applications impractical for the goal of building larger systems as the amount of experiments that must be performed grows exponentially [24, 25].

The randomised benchmarking (RB) protocol [25–30] was developed to be an efficient and scalable procedure which supersedes the conventional methods. It avoids the exponential scaling problem by only allowing for the partial characterisation of gate implementations via a specific gate set known as *unitary 2-designs*, such as the Clifford group. Moreover, by accumulating noise represented by quantum operations in between long sequences of random unitary channels, it extracts an estimate of tomographic data from the targeted gate set, all while being unaffected by SPAM errors. By the original rigorous analyses, in order for RB to work it is required to assume that the noise models are not gate-dependent (i.e. independent of the choice of unitaries) and time-dependent, though work has been done on relaxing these assumptions [31].

As computations involving the Clifford group are manageable, the current RB techniques so far have been restricted to quantum gates within the group. Clifford circuits are important to the quantum com-

[1] Heavily adapted from interim report "Single Qubit Randomised Benchmarking of Non-Clifford Gates" [1].

puting field not only for their considerable usage in fault-tolerant architectures, but as well as being able to be simulated efficiently in classical computing. However, it is known that Clifford gates alone are not enough to achieve universal quantum computation. They require non-Clifford gates, particularly for schemes known as gate injection and magic state distillation [32–34]. The current work that has been done on performing RB with non-Clifford gates is scarce, and mostly only on a specific non-Clifford gate called the T-gate [35–37], which is not a general result. Therefore, it is not clear whether the conventional RB protocol can be performed using the infinite sets of non-Clifford gates of this project that was recently conjectured by Dilkes [38], despite satisfying the condition of being 2-designs.

1.2 The Use of RB Protocol in Current Experiments

The variations of RB protocol, especially Clifford-group RB, have consolidated themselves as the *de facto* method in the quantum technology space on benchmarking and evaluating error rates related to quantum operations. The theoretical framework of Clifford group RB has been rigorously proven, and its robustness and versatility across different quantum architectures have allowed them to be supported and employed in many modern quantum computing experiments [11, 39–41]. An example of this can be seen in a recent breakthrough by Google's ostensible demonstration of quantum supremacy [42]. Their claim to quantum supremacy came from asserting their quantum processor took about 200 seconds to carry out calculations that would take the state of the art supercomputer approximately 10000 years. The Clifford-group RB protocol was used in their experiments, in order to compare with fidelity estimate obtained from their own protocol known as Cross Entropy Benchmarking. It was concluded that the error rates agree in both cases. Their RB curves are taken out of the supplementary information of the main paper Ref.[42] and shown in Fig. 1.1.

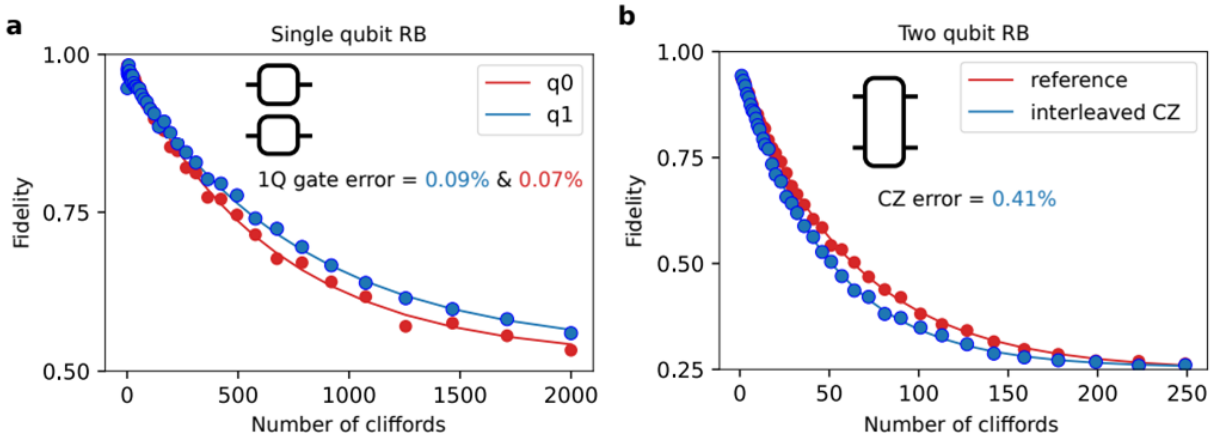

Figure 1.1: **a**) RB data obtained from two separate single qubits. **b**) Two-qubit RB data for a CZ gate performed on the same qubits pair. "interleaved" data refers to a variation of RB where fidelity per gate is obtained instead. Note that in both cases exponential decay curves are seen which are a hallmark of RB. (Both figures take from supplementary information of Ref.[42]).

1.3 Objectives

The objective of this project is to investigate the benchmarking of non-Clifford gates using the single qubit RB protocol. The particular non-Clifford gates are part of infinite families of unitary ensembles known as 2-designs that were conjectured by Dilkes, which built on previous work of the supervisor. The characteristics of these gate sets are numerically simulated with realistic noise models, in order to see if the simulation matches the theory. If these specific non-Clifford gates are viable with the RB protocol, then the characteristic fidelity decay curve will be observable. The sensitivity of the non-Clifford gate sets with respect to the RB protocol will be discussed.

Chapter 2

Theoretical Background

In this chapter, an overview of the mathematical background, notation and tools used in this project are introduced, with the aim of bringing the report to a mathematically self-contained document. Most of the material can be found in standard textbooks and are therefore summarised, with the exception for the research level notations which will be explained concisely and discussed in detail for the reader of this report. For a full overview of quantum information science, the reader is advised to read Ref.[43] which is an excellent textbook for the relevant introductory material.

2.1 Quantum Information

2.1.1 Quantum Bits

In quantum mechanics, the Hilbert space is a complex linear vector space which allows for an inner product, represented by the symbol $\mathcal{H}$ [44]. For an isolated physical system, it is known as its state space such that its complete description is provided by a state vector living on such space. A ket symbol, $|\cdot\rangle$, is commonly used to represent a state vector, which can generally be expressed mathematically as

$$|\psi\rangle = \sum_{i=1}^{n} \alpha_i |\varphi_i\rangle, \tag{2.1}$$

where $|\psi\rangle$ is a linear superposition of n other states. If the set of vectors $\{|\varphi_i\rangle\}_{i=1}^{n}$ is *linearly independent*, such that any linear combination fulfils

$$\alpha_1|\varphi_1\rangle + \ldots + \alpha_n|\varphi_n\rangle = 0 \tag{2.2}$$

only if all coefficients $\{\alpha_i\}_{i=1}^{n}$ equal to zero, then $\{|\varphi_i\rangle\}_{i=1}^{n}$ is said to form a *basis* for $\mathcal{H}$. The number of elements in the basis, n, thus defines the dimension of the Hilbert space.

In classical computation, information is stored in the fundamental unit of bit, represented as 1 or 0. In quantum computation, information is stored in the unit of quantum bit (qubit). For a quantum system, a qubit can generally be represented as a state vector in a linear superposition of two mutually orthonormal states $|1\rangle$ and $|0\rangle$, in a 2-d Hilbert space i.e.

$$\begin{aligned} |\psi\rangle &= \alpha|0\rangle + \beta|1\rangle \\ &= \alpha \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \beta \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \end{aligned} \tag{2.3}$$

where α and β are complex scalars and $|\alpha|^2 + |\beta|^2 = 1$ for a normalised qubit. The set of basis $\{|0\rangle, |1\rangle\}$ is usually known as the *computational basis*.

2.1.2 Density Matrix

However, a single state vector can sometimes be insufficient in describing a quantum system, since the system could be in one of many number of states $|\psi_i\rangle$ with a respective probability p_i, where $\{p_i, |\psi_i\rangle\}$

is said to be an *ensemble of pure states*. For example, imperfections in an experimental device means it will only behave accordingly with a certain probability. As a result, classical randomness can still arise which can contribute additional uncertainty to an experiment, leading to preparation of either state $|\psi_1\rangle$ *or* $|\psi_2\rangle$. Consider a toy example, where an apparatus of a lab experiment is preparing atoms in either states $|\psi_1\rangle$ or $|\psi_2\rangle$ with probabilities p_1 and $p_2 = 1 - p_1$ respectively. When an observable is measured with an operator Q, the expectation value $\langle Q\rangle$ for atoms prepared in $|\psi_1\rangle$ and $|\psi_2\rangle$ will respectively be

$$\langle\psi_1|Q|\psi_1\rangle \text{ and } \langle\psi_2|Q|\psi_2\rangle\,. \tag{2.4}$$

Since $|\psi_1\rangle$ and $|\psi_2\rangle$ have been prepared with probability p_1 and p_2 respectively, an average of the averages must be taken to take into account of this classical randomness, giving

$$\langle Q\rangle = p_1\langle\psi_1|Q|\psi_1\rangle + p_2\langle\psi_2|Q|\psi_2\rangle\,. \tag{2.5}$$

For the convenience of these cases, an alternate formulation which takes into account of this randomness can be used by introducing the *density matrix* or *density operator*. It is mathematically equivalent to the state vector formalism. For a quantum system, the density operator is defined as

$$\rho = \sum_{i=1}^{n} p_i|\psi_i\rangle\langle\psi_i|, \tag{2.6}$$

with the normalisation condition $\sum_i^n p_i = 1$. Compared to the usual quantum superposition, the states $|\psi_i\rangle$ are not required to be orthogonal.

In order to discuss the mathematical properties of a density operator, one must introduce a mathematical tool know as the trace of an operator. The trace is an operation defined as

$$\mathrm{tr}(|u\rangle\langle v|) = \langle v|u\rangle \tag{2.7}$$

for all $|u\rangle, |v\rangle$. The trace is linear, meaning it obeys

$$\begin{aligned}\mathrm{tr}(\alpha|u\rangle\langle v| + \beta|w\rangle\langle x|) &= \alpha\mathrm{tr}(|u\rangle\langle v|) + \beta\mathrm{tr}(|w\rangle\langle x|)\\ &= \alpha\,\langle v|u\rangle + \beta\,\langle x|w\rangle\,,\end{aligned} \tag{2.8}$$

for all α, $\beta \in \mathbb{C}$ and all $|u\rangle$,$|v\rangle$,$|w\rangle$,$|x\rangle \in \mathcal{H}$. It follows that for a measurement operator A acting on a quantum system described by ρ, where A is Hermitian and has a spectral decomposition

$$A = \sum_i \lambda_i|e_i\rangle\langle e_i|. \tag{2.9}$$

Here $|e_i\rangle\langle e_i| = P_i$ is the orthogonal projection operator or projector onto the eigenspace of A, with real and distinct eigenvalues λ_i. The probability of obtaining an observable λ_i can be expressed in a very simple form

$$\begin{aligned}\mathrm{Prob}(\lambda_i) &= \langle e_i|\rho|e_i\rangle\\ &= (\langle e_i|\rho)|e_i\rangle\\ &= \mathrm{tr}(|e_i\rangle\langle e_i|\rho)\\ &= \mathrm{tr}(P_i\rho).\end{aligned} \tag{2.10}$$

These are known as von-Neumann measurements. Finally, the trace has a cylic symmetry property. For all square matrices A, B and C,

$$\begin{aligned}\mathrm{tr}(AB) = \mathrm{tr}(BA) \text{ and}\\ \mathrm{tr}(ABC) = \mathrm{tr}(BCA) = \mathrm{tr}(CAB).\end{aligned} \tag{2.11}$$

With the trace properties now in place, Eq.(2.5) can now be rewritten as the expectation value

$$\begin{aligned}\langle Q\rangle &= p_1\langle\psi_1|Q|\psi_1\rangle + p_2\langle\psi_2|Q|\psi_2\rangle\\ &= p_1\mathrm{tr}(Q|\psi_1\rangle\langle\psi_1|) + p_2\mathrm{tr}(Q|\psi_2\rangle\langle\psi_2|)\\ &= \mathrm{tr}(Qp_1|\psi_1\rangle\langle\psi_1|) + \mathrm{tr}(Qp_2|\psi_2\rangle\langle\psi_2|)\\ &= \mathrm{tr}(Q(p_1|\psi_1\rangle\langle\psi_1| + p_2|\psi_2\rangle\langle\psi_2|))\\ &= \mathrm{tr}(Q\rho),\end{aligned} \tag{2.12}$$

where $\rho = p_1|\psi_1\rangle\langle\psi_1| + p_2|\psi_2\rangle\langle\psi_2|$ is the density operator of the particular system. Generalising ρ to Eq.(2.6), the expectation value is again $\mathrm{tr}(Q\rho)$ for a Hermitian operator Q.

2.1.3 Mixed and Pure States

A density matrix where its exact state is known i.e. consists of only a single projector

$$\rho = |\psi\rangle\langle\psi|, \tag{2.13}$$

is referred to as a *pure state*. Any calculations involving pure states can be done interchangeably between the density operator formalism or the state vector formalism. On the other hand, any state that is not a pure state is a *mixed state*. The density operator of a mixed state cannot be represented by a single projector alone, therefore it has the form of Eq.(2.6), with $i \geq 2$. Note that mixed states cannot be represented by the state vector formalism only. The state $\rho = \frac{I}{2}$, where I is the identity operator, is known as the maximally mixed state. The up state $|0\rangle\langle 0|$ is an example of a pure state, which can be calculated as

$$|0\rangle\langle 0| = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \begin{pmatrix} 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}. \tag{2.14}$$

Physical Example: Faulty Photon Source

In an actual experiment, how might mixed quantum states arise and be created corresponding to a probability distribution? Here is a practical example in which a density operator must be used to describe a physical system.

Information can be encoded within a single particle of light by using their energy, momentum and polarisation etc. in order to create photonic qubits. The presence of a single photon upon creation can have its state be represented by $|1\rangle$, while the absence of a photon can therefore be represented by a vacuum state, $|0\rangle$ [45]. The creation of single photons requires energy to be concentrated into small packets and thus remains a difficult technological challenge. While single photons are used in many quantum communication applications, quantum key distribution protocols in particular require the transmission of single photons through a channel [46]. As a result, the invention for more reliable and efficient single photon sources and detectors remain an active field of research [47]. Even the state of the art single photon sources would not be 100% reliable, meaning they will have a probability p of creating a photon and a probability $1 - p$ of malfunctioning. The success of a photon being generated in this case is thus a source of classical uncertainty. For a single photon source, the state of the system it produces can therefore be described by a mixed state

$$\rho = p|1\rangle\langle 1| + (1-p)|0\rangle\langle 0|. \tag{2.15}$$

2.1.4 POVM Measurements

In some cases, one is only interested in the probabilities of different measurement outcomes rather than the state after measurement of some protocol. Since in this project the state after measurement within the protocol is not considered to be interesting, a generalisation of the projective measurement called the POVM is used instead. Consider an ensemble of positive operators $\{E_m\}$ where they sum to the identity in order to form a normalised probability distribution, i.e. $\sum_m E_m = I$. Such a collection is known as a *Positive Operator Value Measure* (POVM) [48] and each operator E_m is a POVM element of the set which corresponds to a different measurement outcome m. Since the operators E_m are positive, they can be decomposed into $M_m^\dagger M_m$. The probability of obtaining an outcome m when a measurement is made on a quantum state ρ is then given by

$$\begin{aligned} \mathrm{Prob}(m) &= \mathrm{tr}(M_m \rho M_m^\dagger) \\ &= \mathrm{tr}(M_m^\dagger M_m \rho) \\ &= \mathrm{tr}(E_m \rho). \end{aligned} \tag{2.16}$$

Note that in this instance the operators $\{E_m\}$ do not have to be orthogonal to each other. Therefore, there is no limit to the amount of elements a set of POVM can have which correspond to a number of

different outcomes [49]. So long as the set of operators satisfy the completeness relation $\sum_m M_m^\dagger M_m = I$, they correspond to a valid set of operations leading to a collection of post-measurement states and a well-defined probability distribution. In the case where $\{E_m\}$ do form a set of orthogonal measurement operators P_m, with P_m being projectors which satisfy $P_m P_{m'} = \delta_{mm'} P_m$ and $\sum_m P_m = I$, then the rules for von-Neumann measurements are recovered as in section 2.1.2.

2.1.5 Quantum Circuits

In the quantum circuit model of quantum computation, information processing is represented by a diagram called the quantum circuit. In order to carry out a computation, a series of unitary transformations are applied onto a prepared state in order to represent its evolution. Analogous to computation on a classical computer, the unitary transformations are known as quantum gates, individually separated by a horizontal "wire" on the diagram. In this section some commonly used quantum gates will be defined, and a diagrammatic example of a quantum circuit will be provided.

The Pauli matrices are a set of quantum gates in the computational basis given by

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \; X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \; Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \; Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{2.17}$$

Together the Pauli matrices form the Pauli group $\mathcal{P} = \{I, X, Y, Z\}$. The Hadamard gate ($H$) and the phase gate ($S$) are given by

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \; S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}. \tag{2.18}$$

For a single qubit, quantum gates are (2×2) matrices. I is the identity matrix. X induces a bit flip, changing $|0\rangle$ to $|1\rangle$ and vice versa. Z induces a phase flip, leaving $|0\rangle$ untouched and maps $|1\rangle$ to $-|1\rangle$. Y induces a bit flip and a phase flip since $Y = iXZ$, mapping $|0\rangle$ to $i|1\rangle$ and $|1\rangle$ to $-i|0\rangle$. H induces the basis states to a superposition, mapping $|0\rangle$ to $|+\rangle = \frac{|0\rangle+|1\rangle}{\sqrt{2}}$ and $|1\rangle$ to $|-\rangle = \frac{|0\rangle-|1\rangle}{\sqrt{2}}$. A single qubit quantum circuit is shown in Fig. 2.1.5. The straight horizontal line connecting the quantum gates propagates the qubit, with time going from left to right. $|\psi\rangle$ represents the prepared input state and $|\psi'\rangle$ represents the output state after a measurement. The meter box represents a projective measurement in $\{0, 1\}$ basis.

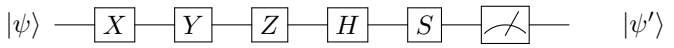

Figure 2.1: An example of a quantum circuit for a single qubit involving the aforementioned gates.

2.2 Quantum Operations

From here onwards both terms map and channel will be used interchangeably. General and non-unitary channels are denoted by calligraphic letters such as $\mathcal{E}$ and $\mathcal{C}$ whereas $\mathcal{U}$ is reserved for unitary channels. In an open quantum system, a state evolves under a quantum channel $\mathcal{E}$, which maps a vector linearly over its space. Under such a dynamical process, $\mathcal{E}$ is called a superoperator and it transforms an initial state ρ into a final state ρ' i.e.

$$\rho \to \rho' = \mathcal{E}(\rho). \tag{2.19}$$

The channel $\mathcal{E}$ is generally a linear, *completely positive trace preserving* (CPTP) map such that it must fulfil the following requirements[1] [50]:

1. $\mathcal{E}$ operates linearly in density operator i.e. for a state such as Eq.(2.6),

$$\mathcal{E}\left(\sum_{i=1}^{n} p_i |\psi_i\rangle\langle\psi_i|\right) = \sum_{i=1}^{n} p_i \mathcal{E}(|\psi_i\rangle\langle\psi_i|). \tag{2.20}$$

[1] Heavily adapted from interim report "Single Qubit Randomised Benchmarking of Non-Clifford Gates" [1].

2. $\mathcal{E}$ is trace preserving (TP) such that $\mathrm{tr}(\mathcal{E}(\rho)) = \mathrm{tr}(\rho) = 1$. This ensures that the state of the quantum system does not just disappear.

3. $\mathcal{E}(\rho)$ is positive semi-definite (positive), meaning it is Hermitian and its eigenvalues are strictly positive. For a positive ρ, it follows that $\mathcal{E}(\rho)$ must also be positive.

4. $\mathcal{E}$ is completely positive (CP). A composite system comprised of subsystems ρ_A and ρ_B can be described by a joint state ρ_{AB}. It follows that it is possible for some superoperators acting on its reduced state $\mathcal{E}(\rho_A)$ to return negative eigenvalues, despite never return any negative eigenvalues when acted on its full state $\mathcal{E}(\rho_{AB})$. Thus, a superoperator is CP if it never returns any negative eigenvalues when it acts on a full system or even on a sub-system alone.

The conditions $(1-4)$ ensure that the density operator ρ stays normalised when it is mapped to ρ' by $\mathcal{E}$. On top of that, condition (4) also allows for the extended mapping of density operators to density operators on a larger composite system. This is a crucial requirement for any acceptable map, since *a priori* the possibility of a principle system being initially entangled with some remote isolated system cannot be ruled out [51].

In order for a quantum state to evolve physically, its dynamical evolution must be described by a CPTP map so that it can be created in reality. Thus, a quantum channel is not only equivalent to a quantum gate, but also to a noise map under this definition. In general, the mathematical representation of CPTP maps comes in many form. The two that are mainly used in this report are the Kraus representation (also known as operator-sum) and a variation of the Liouville (also known as superoperator) representation which will be covered below.

2.2.1 Operator-sum/Kraus Representation

Provided that a superoperator is CPTP, it has an operator-sum decomposition and thus can be written in its Kraus representation[2] [52, 53]

$$\mathcal{E}(\rho) = \sum_i K_i \rho K_i^\dagger, \tag{2.21}$$

where $\{K_i\}$ are called a set of Kraus operators. Since Kraus operators are required to be TP, it can be shown that they satisfy

$$\sum_i K_i^\dagger K_i = I, \tag{2.22}$$

which is equivalent to the completeness relation [43]. By the same token, a superoperator $\mathcal{E}$ is a CPTP map if and only if it has a Kraus decomposition with respect to Eq.(2.22). Furthermore, there can be at most d^2 Kraus operators representing a CPTP map which acts on d-dimensional quantum states. The Kraus representation is a powerful mathematical framework, since it describes the dynamical process of a particular principal system without having to explicitly consider the attributes of its surrounding environment. They are all incorporated within the Kraus operators K_i. The characterisation of a transformation by the operators is therefore intrinsic, since they only act on the principal system. Within the Kraus representation however, there is a requirement that there are no correlation between the system and the environment initially, otherwise non-completely positive maps can arise [54]. With all things considered above, the Kraus operators are therefore important quantities which are usually determined by experimentalists, i.e. by QPT [55].

Ideal and Noisy Unitary Channel Composition in Kraus Representation

For this report only single qubit systems are concerned, therefore quantum channels are acting in a 2-dimensional Hilbert space $\mathcal{H} = \mathbb{C}^2$. For a density operator ρ evolving under a unitary operation (gate) U, its action can be described by a superoperator $\mathcal{U}$, denoted as

$$\mathcal{U}(\rho) = U\rho U^\dagger, \tag{2.23}$$

[2] Heavily adapted from interim report "Single Qubit Randomised Benchmarking of Non-Clifford Gates" [1].

where such a unitary channel can be thought as a Kraus operation with a single Kraus operator U. The adjoint channel is then $\mathcal{U}^\dagger(\rho) = U^\dagger \rho U$ and m applications are $\mathcal{U}^m(\rho) = U...U\rho U^\dagger...U^\dagger = U^m \rho U^{m\dagger}$. The noisy implementation of the ideal operation $\mathcal{U}$, denoted as $\tilde{\mathcal{U}}$, is approximated as the ideal target operation U composed with a CPTP map $\mathcal{D}$ as a noise map (not necessarily unitary) such that $\tilde{\mathcal{U}} = \mathcal{D} \circ \mathcal{U}$, with "$\circ$" denoting the channel composition. In general, a sequential composition of n channels is written as

$$\begin{aligned} \bigcirc_{i=1}^{n} \mathcal{E}_i(\rho) &= \mathcal{E}_n(...\mathcal{E}_2(\mathcal{E}_1(\rho))...) \\ &= \mathcal{E}_n \circ \cdots \circ \mathcal{E}_1(\rho), \end{aligned} \tag{2.24}$$

where $\mathcal{E}_i$ can be a non-unitary superoperator, applied from right to left.

A common noise channel is the *depolarising channel*, which either has the probability $(1-p)$ of doing nothing, or a probability p of mapping an input state to a maximally mixed state $\frac{I}{2}$. It has the form

$$\begin{aligned} \mathcal{D}(\rho) &= (1-p)\rho + p\frac{I}{2} \\ &= (1-p)\rho + \frac{p}{3}(X\rho X + Y\rho Y + Z\rho Z). \end{aligned} \tag{2.25}$$

Here a common Kraus decomposition of the depolarising channel is shown, with Kraus operators $K_i \in \{\sqrt{1-p}I, \sqrt{\frac{p}{3}}X, \sqrt{\frac{p}{3}}Y, \sqrt{\frac{p}{3}}Z\}$. A noisy unitary operation with $\mathcal{D}$ then is explicitly shown as

$$\begin{aligned} \tilde{\mathcal{U}}(\rho) &= \mathcal{D} \circ \mathcal{U}(\rho) \\ &= \mathcal{D}(U\rho U^\dagger) \\ &= (\sqrt{1-p})^2 U\rho U^\dagger + \left(\sqrt{\frac{p}{3}}\right)^2 (XU\rho U^\dagger X + YU\rho U^\dagger Y + ZU\rho U^\dagger Z) \\ &= (1-p)\mathcal{U}(\rho) + \frac{p}{3}[X\mathcal{U}(\rho)X + Y\mathcal{U}(\rho)Y + Z\mathcal{U}(\rho)Z] \end{aligned} \tag{2.26}$$

For a noisy sequential composition of channels, it can be represented with one or two noise maps composed with every single channel within the sequence, i.e. a noisy version of Eq.(2.24) is written as

$$\begin{aligned} \bigcirc_{i=1}^{n} \tilde{\mathcal{E}}_i(\rho) &= \bigcirc_{i=1}^{n} [\mathcal{F}_i \circ \mathcal{D}_i \circ \mathcal{E}_i](\rho) \\ &= \mathcal{F}_n(\mathcal{D}_n(\mathcal{E}_n(...\mathcal{F}_1(\mathcal{D}_1(\mathcal{E}_1(\rho)))...))) \\ &= \mathcal{F}_n \circ \mathcal{D}_n \circ \mathcal{E}_n \circ \cdots \circ \mathcal{F}_1 \circ \mathcal{D}_1 \circ \mathcal{E}_1(\rho) \\ &= \tilde{\mathcal{E}}_n \circ \cdots \circ \tilde{\mathcal{E}}_1(\rho). \end{aligned} \tag{2.27}$$

2.2.2 Superoperator/Liouville Representation

The Liouville (also known as superoperator) representation of quantum channels [15] is used intensively in this project, due to its convenience in manipulating and constructing a composition of quantum operations. In this formalism, a density operator ρ on a d-dimensional Hilbert space is vectorised to being a vector $|\rho\rangle\!\rangle$ on a d^2-dimensional *Hilbert-Schmidt* space. Specifically, the space of $(d \times d)$ complex matrices are now being represented as $(d^2 - 1)$ vectors. For a given trace-orthonormal basis operators $\mathcal{B} = \{B_0, \ldots, B_{d^2}\}$ in the operator space $\mathcal{H}_{d^2}$, a density operator can be written as a linear combination of a scalar multiplying each basis vector over all basis vectors i.e.

$$\rho = \sum_{k=1}^{d^2} \langle B_k, \rho \rangle B_k. \tag{2.28}$$

The scalar $\langle B_k, \rho \rangle$ is the Hilbert-Schmidt inner product, defined as $\langle A, B \rangle = \mathrm{tr}(A^\dagger B)$, hence in this case $\langle B_k, \rho \rangle = \mathrm{tr}(B_k^\dagger \rho)$ with respect to the k-th basis operator. The orthonormality condition is thus defined by the Hilbert-Schmidt inner product as $\mathrm{tr}(B_j^\dagger B_k) = \delta_{jk}$. The density operator ρ can now be represented by a column vector $|\rho\rangle\!\rangle \in \mathbb{C}^{d^2}$, with its k-th vector element given by $\langle B_k, \rho \rangle$. Note that $\langle\!\langle \rho| = (|\rho\rangle\!\rangle)^\dagger$. For

a single qubit with computational basis $\{|0\rangle\langle 0|, |1\rangle\langle 0|, |0\rangle\langle 1|, |1\rangle\langle 1|\}$, the vectorisation $\rho \to |\rho\rangle\rangle$ is done by stacking the entries in the density matrix column-by-column into a column vector. For example,

$$\rho = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \to |\rho\rangle\rangle = \begin{pmatrix} a \\ c \\ b \\ d \end{pmatrix}. \tag{2.29}$$

To see why this is the case, each vector entry can be computed according to the Hilbert-Schmidt inner product with the relevant computational basis

$$\mathrm{tr}(|0\rangle\langle 0|\rho) = \mathrm{tr}\left(\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}\rho\right) = a \tag{2.30}$$

$$\begin{aligned} \mathrm{tr}(|1\rangle\langle 0|\rho) &= c \\ \mathrm{tr}(|0\rangle\langle 1|\rho) &= b \\ \mathrm{tr}(|1\rangle\langle 1|\rho) &= d. \end{aligned} \tag{2.31}$$

Similarly, a measurement can be represented by a row vector $\langle\langle E|$ with its k-th elements this time given by $\mathrm{tr}(E^\dagger B_k)$. The Born rule in Liouville representation can then be given as an overlap of the vectorised row and column vectors as

$$\langle\langle E|\rho\rangle\rangle = \mathrm{tr}(E^\dagger \rho), \tag{2.32}$$

which gives the probability of observing a projective measurement E for a prepared quantum state ρ.

As a consequence, the superoperator representation allows for a quantum operation (linear CPTP map) $\mathcal{E}$ that maps a density operator ρ to another density operator $\rho' = \mathcal{E}(\rho)$ to be represented as a matrix $\boldsymbol{\mathcal{E}}$ with dimension $(d^2 \times d^2)$ as (with a slight abuse of notation) [56]

$$\boldsymbol{\mathcal{E}}|\rho\rangle\rangle = |\mathcal{E}(\rho)\rangle\rangle \tag{2.33}$$

for all ρ, with entries

$$\begin{aligned} \boldsymbol{\mathcal{E}}_{kl} &= \mathrm{tr}(B_k^\dagger \mathcal{E}(B_l)) \\ &= \langle B_k, \mathcal{E}(B_l)\rangle \\ &= \langle\langle B_k|\boldsymbol{\mathcal{E}}|B_l\rangle\rangle \end{aligned} \tag{2.34}$$

where $\boldsymbol{\mathcal{E}} \in \mathbb{C}^{d^2 \times d^2}$ and such a channel is a unique matrix. Quantum channels in such representation will hereby be represented by bold calligraphic letters $\boldsymbol{\mathcal{E}}, \boldsymbol{\mathcal{D}}$ etc with $\boldsymbol{\mathcal{U}}$ again reserved for unitary operations. The Liouville representation is useful and convenient for quantum information processing for a few particular reasons:

1. Unitary operators are now contained within a single matrix. Unitary transformation of a density operator ρ to ρ' by Eq.(2.23) is now done as $|\rho'\rangle\rangle = \boldsymbol{\mathcal{U}}|\rho\rangle\rangle$ by a single Liouville map $\boldsymbol{\mathcal{U}}$, whose elements are calculated by $\boldsymbol{\mathcal{U}}_{kl} = \langle\langle B_k|\boldsymbol{\mathcal{U}}|B_l\rangle\rangle$ with any normalised operator basis $\{B_0, ..., B_{d^2}\}$.

2. Channel composition in general can be represented by just matrix multiplication, without having to act on the state first. As opposed to Kraus representation where operations are done under the conjugation of a state ρ, where the channel composition of two channels $\mathcal{E}_1$ and $\mathcal{E}_2$ are $\mathcal{E}'(\rho) = \mathcal{E}_2(\mathcal{E}_1(\rho))$ according to Eq.(2.21). In Liouville representation this corresponds to simply $\boldsymbol{\mathcal{E}}' = \boldsymbol{\mathcal{E}}_2\boldsymbol{\mathcal{E}}_1$.

3. Measurements with POVMs are done with just the Born rule. Similar to Eq.(2.28), a single POVM element can also be expanded by the basis operators $\{B_0, ..., B_{d^2}\}$ as $E = \sum_k \langle E, B_k\rangle B_k^\dagger$, and identify $\langle E, B_k\rangle$ being the elements of the row vector $\langle\langle E|$. A measurement after a single channel operation $\boldsymbol{\mathcal{E}}$ on a density operator can thus be written as $\langle\langle E|\boldsymbol{\mathcal{E}}|\rho\rangle\rangle$.

2.2.3 Pauli-Liouville Representation

When the operator basis in Liouville representation is chosen specifically to be the Pauli basis, it is known as the Pauli-Liouville representation. The numerical simulation of this project will mainly work in this representation, for a single qubit.

Let the operator basis $\mathcal{B} = \{B_0, \ldots, B_{d^2}\}$ be chosen as the rescaled Pauli group where $\mathcal{P} \to \mathcal{P}/\sqrt{2}$, ensuring that the basis is normalised. By standard convention, B_0 is fixed to be $I/\sqrt{2}$, as it sets the trace-preserving condition and the condition that the first element has a non-zero trace with all the other elements traceless. The subsequent basis elements are set to be $X/\sqrt{2}$, $Y/\sqrt{2}$ and $Z/\sqrt{2}$ respectively. Since the Pauli operators are Hermitian, such a choice of basis ensures that the elements of any Pauli-Liouville matrix to be real as well as its adjoint channel being also its transpose, i.e. $\boldsymbol{\mathcal{E}}^\dagger = \boldsymbol{\mathcal{E}}^{\mathrm{T}}$. With respect to the Pauli basis, the column vector $|\rho\rangle\!\rangle$ now has elements $\rho_j = \langle \mathcal{P}_j, \rho \rangle$. For example, a down state is vectorised as

$$|1\rangle\langle 1| = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \to |\downarrow\rangle\!\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 0 \\ 0 \\ -1 \end{pmatrix} \tag{2.35}$$

since

$$\begin{aligned} \rho_1 &= \mathrm{tr}\left(\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}\right) = \frac{1}{\sqrt{2}} \\ \rho_2 &= \mathrm{tr}\left(\frac{1}{\sqrt{2}}\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}\right) = 0 \\ \rho_3 &= \mathrm{tr}\left(\frac{1}{\sqrt{2}}\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}\right) = 0 \\ \rho_4 &= \mathrm{tr}\left(\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}\right) = -\frac{1}{\sqrt{2}}. \end{aligned} \tag{2.36}$$

A quantum channel $\boldsymbol{\mathcal{E}}$ is again a single 4×4 matrix, with entries $\mathrm{tr}(\mathcal{P}_k^\dagger \mathcal{E}(\mathcal{P}_l))$ according to Eq.(2.34) and $\mathcal{E}(\mathcal{P}_l)$ according to Eq.(2.21). For example, a channel $\boldsymbol{\mathcal{E}}$ that applies the X gate to the state ρ by Kraus operation $\mathcal{E}(\rho) = X\rho X^\dagger$ now has the Pauli-Liouville vectorised form

$$\boldsymbol{\mathcal{E}} = \frac{1}{2}\begin{pmatrix} \mathrm{tr}(I\mathcal{E}(I)) & \mathrm{tr}(I\mathcal{E}(X)) & \mathrm{tr}(I\mathcal{E}(Y)) & \mathrm{tr}(I\mathcal{E}(Z)) \\ \mathrm{tr}(X\mathcal{E}(I)) & \mathrm{tr}(X\mathcal{E}(X)) & \mathrm{tr}(X\mathcal{E}(Y)) & \mathrm{tr}(X\mathcal{E}(Z)) \\ \mathrm{tr}(Y\mathcal{E}(I)) & \mathrm{tr}(Y\mathcal{E}(X)) & \mathrm{tr}(Y\mathcal{E}(Y)) & \mathrm{tr}(Y\mathcal{E}(Z)) \\ \mathrm{tr}(Z\mathcal{E}(I)) & \mathrm{tr}(Z\mathcal{E}(X)) & \mathrm{tr}(Z\mathcal{E}(Y)) & \mathrm{tr}(Z\mathcal{E}(Z)) \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}. \tag{2.37}$$

Applying this channel to the down state $|\downarrow\rangle\!\rangle$ from before gives the vectorised up state $|\uparrow\rangle\!\rangle = \frac{1}{\sqrt{2}}(1\ 0\ 0\ 1)^{\mathrm{T}}$, which is as expected in any formalism.

General CPTP Map in Pauli-Liouville Representation

In Pauli-Liouville representation, all TP maps for a single qubit can be expressed in a nice form of [57, 58]

$$\boldsymbol{\Phi} = \boldsymbol{\mathcal{U}}\boldsymbol{\Lambda}\boldsymbol{\mathcal{V}} \tag{2.38}$$

where $\boldsymbol{\mathcal{U}}$ and $\boldsymbol{\mathcal{V}}$ are unitary matrices and $\boldsymbol{\Lambda}$ has a general form of

$$\boldsymbol{\Lambda} = \begin{pmatrix} 1 & 0 \\ \boldsymbol{\Lambda}_n & \boldsymbol{\Lambda}_u \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ t_1 & \lambda_1 & 0 & 0 \\ t_2 & 0 & \lambda_2 & 0 \\ t_3 & 0 & 0 & \lambda_3 \end{pmatrix}. \tag{2.39}$$

Here $\boldsymbol{\Lambda}_n$ is a 3×1 column vector and $\boldsymbol{\Lambda}_u$ is a 3×3 submatrix. They are both respectively known as the non-unital and unital part of Λ. A map is said to be unital if $\boldsymbol{\Lambda}_n = 0$. The matrices $\boldsymbol{\mathcal{U}}$ and $\boldsymbol{\mathcal{V}}$ are constructed

such that it applies a general small angle rotation composed of a Z rotation $R_Z = \exp(-i\theta Z/2)$ followed by an X rotation $R_X = \exp(-i\theta X/2)$ followed by another Z rotation [59]. In Pauli-Liouville form, they are

$$\boldsymbol{\mathcal{U}}, \boldsymbol{\mathcal{V}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta_1 & \sin\theta_1 & 0 \\ 0 & -\sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \cos\theta_2 & \sin\theta_2 \\ 0 & 0 & -\sin\theta_2 & \cos\theta_2 \end{pmatrix} * \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta_3 & \sin\theta_3 & 0 \\ 0 & -\sin\theta_3 & \cos\theta_3 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{2.40}$$

The fact that $\boldsymbol{\Lambda}_{11} = 1$ is a consequence of the TP condition of the map, which can be shown using the Kraus representation [56]. Let $\boldsymbol{\Lambda}$ be a Pauli-Liouville channel again be described by Eq.(2.34). Since it is TP, it follows that $\mathcal{E}$ has an operator-sum decomposition

$$\begin{aligned} \boldsymbol{\Lambda}_{kl} &= \mathrm{tr}(P_k^\dagger \mathcal{E}(P_l)) \\ &= \mathrm{tr}\left(P_k^\dagger \sum_m K_m P_l K_m^\dagger\right) \\ &= \mathrm{tr}\left(P_l \sum_m K_m^\dagger P_k^\dagger K_m\right), \end{aligned} \tag{2.41}$$

where the third line is due to the trace being cyclic and linear in its arguments. By fixing the first element of the operator basis to be $B_1 = P_1 = I/\sqrt{2}$ implies

$$\begin{aligned} \boldsymbol{\Lambda}_{11} &= \mathrm{tr}\left(P_1 \sum_m K_m^\dagger P_1^\dagger K_m\right) \\ &= \mathrm{tr}\left(\frac{1}{2} \sum_m K_m^\dagger K_m\right). \end{aligned} \tag{2.42}$$

Following the TP condition of Kraus operators from Eq.(2.22)

$$\boldsymbol{\Lambda}_{11} = \mathrm{tr}\left(\frac{I}{2}\right) = 1. \tag{2.43}$$

In general, if $\boldsymbol{\Lambda}_{11} < 1$ then it is known as a trace-decreasing map. In addition, for such a map to also be CP, the diagonal elements $\lambda_1, \lambda_2, \lambda_3$ of $\boldsymbol{\Lambda}$ must satisfy [60]

$$|1 \pm \lambda_3| \geq |\lambda_1 \pm \lambda_2|. \tag{2.44}$$

With the aforementioned constraints, an arbitrary CPTP map $\boldsymbol{\Phi}$ can be constructed. Such a map in this particular representation has the unique property of being able to transform into a quantum depolarising channel under an action called twirl by a unitary 2-design, both of which are further discussed in the next section. Hence, it will be used as the main noise channel for the randomised benchmarking protocol simulation of this project.

Ideal and Noisy Unitary Evolution in Pauli-Liouville Representation

The unitary evolution of a density operator $|\rho\rangle\!\rangle \to |\rho'\rangle\!\rangle$ is described by $|\rho'\rangle\!\rangle = \boldsymbol{\mathcal{U}}|\rho\rangle\!\rangle$, with an adjoint channel $\boldsymbol{\mathcal{U}}^\dagger$ such that $\boldsymbol{\mathcal{U}}^\dagger|\rho'\rangle\!\rangle = |\rho\rangle\!\rangle$. For m applications of the same unitary channel, it is described by $|\rho'\rangle\!\rangle = \boldsymbol{\mathcal{U}}..\boldsymbol{\mathcal{U}}\boldsymbol{\mathcal{U}}|\rho\rangle\!\rangle = \boldsymbol{\mathcal{U}}^m|\rho'\rangle\!\rangle$. The noisy implementation of $\boldsymbol{\mathcal{U}}$, again denoted with tilde as $\tilde{\boldsymbol{\mathcal{U}}}$, is implemented as a general CPTP map $\boldsymbol{\mathcal{D}}$ composed with itself such that $\tilde{\boldsymbol{\mathcal{U}}} = \boldsymbol{\mathcal{D}} \circ \boldsymbol{\mathcal{U}} = \boldsymbol{\mathcal{D}}\boldsymbol{\mathcal{U}}$, where channel composition is now simply a matrix multiplication. In general, a sequential composition of n different channels is represented as

$$\boldsymbol{\mathcal{U}}_n \circ ... \circ \boldsymbol{\mathcal{U}}_2 \circ \boldsymbol{\mathcal{U}}_1 = \boldsymbol{\mathcal{U}}_n ... \boldsymbol{\mathcal{U}}_2 \boldsymbol{\mathcal{U}}_1, \tag{2.45}$$

applied from right to left and without having to apply onto a quantum state first. A noisy version of such sequence of channels would simply be

$$\begin{aligned} \boldsymbol{\mathcal{D}}_n \circ \boldsymbol{\mathcal{U}}_n \circ ... \circ \boldsymbol{\mathcal{D}}_2 \circ \boldsymbol{\mathcal{U}}_2 \circ \boldsymbol{\mathcal{D}}_1 \circ \boldsymbol{\mathcal{U}}_1 &= \boldsymbol{\mathcal{D}}_n \boldsymbol{\mathcal{U}}_n ... \boldsymbol{\mathcal{D}}_2 \boldsymbol{\mathcal{U}}_2 \boldsymbol{\mathcal{D}}_1 \boldsymbol{\mathcal{U}}_1 \\ &= \tilde{\boldsymbol{\mathcal{U}}}_n ... \tilde{\boldsymbol{\mathcal{U}}}_2 \tilde{\boldsymbol{\mathcal{U}}}_1, \end{aligned} \tag{2.46}$$

where $\{\boldsymbol{\mathcal{D}}_i\}_{i=1...n}$ are a set of different general CPTP maps.

2.3 t-Designs and Clifford Gates

Randomness as a resource is both useful and important in classical and quantum physics, laying foundations in computation, simulation and communication [61]. The practical usefulness of randomness has been demonstrated in the past, with an example of being used to develop an algorithm which checks probabilistically whether a number is composite or prime [62]. However, perfect randomness is expensive since the computational resources required to produce them are immense. In fact, perfect randomness can rarely be achieved, since the outputs of perfect random data must have equal weights, which can usually only be done in laboratories under strict constraints [63].

In order to get over the resource intensiveness of obtaining perfect randomness, one usually resorts to achieving good pseudorandomness. If the outputs from machine A with perfect randomness and machine B with pseudorandomness are indistinguishable from one another in any reasonable amount of time, then the pseudorandomness of machine B is considered good [64]. In quantum computation, the randomisation of all unitary matrices over the unitary group is often required. In such cases problems tend to arise since the size of the group could be infinite. As a consequence, *quantum t-designs* are often used since in most cases they allow for perfect random operations to be replaced, and demonstrate that in general quantum dynamics are indistinguishable from truly random processes [65]. They have various practical applications such pseudorandomness generation [61], quantum cryptography [66], QPT [67] and most notably randomised benchmarking [68].

A quantum t-design is a probability distribution over a set of mathematical objects of choice, such that its statistical moments for a degree t or less, is indistinguishable from those taken over the whole set with respect to a uniform measure (probability distribution). Specifically, averaging any degree t polynomial function over such a finite ensemble is equivalent to their average over a uniform measure. In addition, any t-design with degree t is also a t-design of degree less than t, since the set of polynomials with degree less than t forms a subset of the set of degree t polynomials [69]. For example, a t-design is also a $(t-1)$-design. If the mathematical objects are chosen to be an ensemble of pure quantum states or unitary matrices, then they are called *state t-designs* or *unitary t-design* respectively. Here the uniform measure is the Haar measure, which is a unique uniform probability distribution over either all pure quantum states or all unitary matrices. For the purpose of this project, only unitary t-designs will be reviewed since unitary 2-designs are the relevant gate set to the randomised benchmarking simulation of the project.

2.3.1 Unitary t-Designs

A finite ensemble $\{U_i\}_{i=1\ldots N} \in \mathrm{U}(d)$ of unitary operators on Hilbert space $\mathcal{H} = \mathbb{C}^d$ is defined concisely in Ref.[70] to be a unitary t-design in d-dimensions, if for every polynomial $P_{(t,t)}(U)$ which is homogeneous of degree at most t in each variable of all $2d^2$ variables i.e. $P_{(t,t)}(U)$ is a function of degree at most t in the matrix elements of U and at most t in the complex conjugates of those matrix elements, $P_{(t,t)}(U) = P_{(t,t)}(U^j_k, U^{*j}_k)$, it satisfies

$$\frac{1}{N}\sum_{i=1}^{N} P_{(t,t)}(U_i) = \int_{\mathrm{U}(d)} dU\, P_{(t,t)}(U). \tag{2.47}$$

Here the integral over $\mathrm{U}(d)$ is done with respect to the Haar measure. With a finite selection of unitary matrices that is the unitary t-design, the properties of the characteristic of the whole unitary group $\mathrm{U}(d)$ under Haar measure can be reproduced. An equivalent statement to Eq.(2.47) is

$$\frac{1}{N}\sum_{i=1}^{N} U_i^{\otimes t} \otimes (U_i^\dagger)^{\otimes t} = \int_{\mathrm{U}(d)} dU U^{\otimes t} \otimes (U^\dagger)^{\otimes t}, \tag{2.48}$$

with the unitary matrices in general have the parametrised form

$$U(\theta, \Phi, \eta) = \begin{pmatrix} e^{i\Phi}\cos\theta & -e^{i\eta}\sin\theta \\ e^{-i\eta}\sin\theta & e^{-i\Phi}\cos\theta \end{pmatrix} = \begin{pmatrix} u & v \\ v^* & u^* \end{pmatrix}, \tag{2.49}$$

with the Haar measure as

$$dU = \frac{\sin 2\theta}{4\pi^2} d\theta d\Phi d\eta, \tag{2.50}$$

which is normalised so that $\int_{\mathrm{U}(d)} dU = 1$. Here $U^{\otimes t}$ refers to the tensor product $\otimes$ of U with itself t times.

2.3.2 Unitary 2-Designs

A unitary 2-design is, very simply, just a unitary t-design with degree $t = 2$. The 2-designs have a useful property where the sampling from such a finite ensemble $\{U_i\}_{i=1...N}$ can replace the sampling from the Haar measure over the whole unitary group $\mathrm{U}(d)$. This implies that the resources and computational complexity needed to implement randomisation over unitary matrices can be significantly reduced. Another definition which was proven in Ref.[71] to be identical to Eq.(2.48) is that results of twirling over the unitary group through all possible unitary matrices and the results of twirling over all element within a 2-design are equivalent. Specifically, the *twirl* of all unitary matrices over the unitary group sampled from the Haar measure for any channel $\mathcal{E}$ acting on any quantum state ρ is defined to be

$$\int_{\mathrm{U}(d)} dU(U^\dagger \mathcal{E}(U\rho U^\dagger)U) \equiv \int_{\mathrm{U}(d)} dU \mathcal{U}^\dagger \circ \mathcal{E} \circ \mathcal{U}(\rho). \tag{2.51}$$

Here it can clearly be seen that the integrand is a polynomial of degree 2 with $U^{\otimes 2} = U \otimes U$. It therefore gives

$$\frac{1}{N}\sum_{i=1}^{N}(U_i^\dagger \mathcal{E}(U_i \rho U_i^\dagger)U_i) = \int_{\mathrm{U}(d)} dU(U^\dagger \mathcal{E}(U\rho U^\dagger)U) \tag{2.52}$$

by Eq.(2.48), with the twirl now acts over all elements of unitary 2-designs. The number of unitary matrices to twirl over is now equal to just the size of the 2-design, and this is important for measuring the average gate fidelity of a channel within randomised benchmarking. By some Representation Theory arguments which are beyond the scope of this project, it turns out that the action of the twirl with unitary 2-designs transform any arbitrary quantum channel in Pauli-Liouville representation into the form of

$$\boldsymbol{\mathcal{D}}_p = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & p & 0 & 0 \\ 0 & 0 & p & 0 \\ 0 & 0 & 0 & p \end{pmatrix}, \tag{2.53}$$

which is the Pauli-Liouville representation of a depolarising channel, similar to Eq.(2.25), where p is the depolarising parameter. For the precise proof and arguments, the readers are referred to [60, 72]. This is the precise reason why matrices of the form Eq.(2.38) are used as the noise map for the RB simulation of this project. Eq.(2.53) allows for the extraction of average gate fidelity through randomised benchmarking, and is further introduced and discussed in section 2.4.2.

2.3.3 The Frame Potential

Given a particular set of unitary matrices $\mathcal{D} = \{U_i\}_{i=1...N}$, the straightforward way to check if it is a 2-design would be to compute and compare both side of Eq.(2.48). However, a simpler method of doing so would be to use the *frame potential*, defined as [72]

$$\Phi(\mathcal{D}) = \frac{1}{N^2}\sum_{U_i, U_j \in \mathcal{D}} |\mathrm{tr}(U_i^\dagger U_j)|^4 \tag{2.54}$$

The set $\mathcal{D}$ is a 2-design if and only if $\Phi(\mathcal{D}) = 2$. An example of a 2-design is the minimum single qubit 2-design, given by

$$\mathcal{D}_{12} = \{I, X, Y, Z, I + i(\pm X \pm Y \pm Z)\}, \tag{2.55}$$

which can be confirmed with the frame potential returning $\Phi(\mathcal{D}_{12}) = 2$.

2.3.4 The Clifford Group

A group $\mathcal{G}$ is a set of element $\{g_i\} \in \mathcal{G}$ with a group operation "$\cdot$" such that they obey four mathematical axioms [73]:

1. (Closure) There exists some operator elements $g_1, g_2 \in \mathcal{G}$ where $g_1 \cdot g_2 = g_3,\ g_3 \in \mathcal{G},\ \forall\ g \in \mathcal{G}$.

2. (Associativity) The operators are associative; for $g_1, g_2, g_3 \in \mathcal{G}$, $(g_1 \cdot g_2) \cdot g_3 = g_1 \cdot (g_2 \cdot g_3)$.

3. (Identity) There exists an identity $I \in \mathcal{G}$ where $I \cdot g = g \cdot I = g,\ \forall\ g \in \mathcal{G}$.

4. (Inversibility) There exists an inverse operator element $g^{-1} \in \mathcal{G},\ \forall\ g \in \mathcal{G}$ where $g^{-1} \cdot g = g \cdot g^{-1} = I$.

The normalizer of a subset S of a group $\mathcal{G}$ is defined as

$$N_G(S) = \{\forall\ g \in \mathcal{G} \mid S = gSg^{-1}\}. \tag{2.56}$$

In this report, only the Pauli group and the Clifford group will be relevant. The n-qubit Clifford group, denoted as $\mathcal{C}_n$, can now be defined as a set of unitary operators which is the normalizer of the Pauli group up to overall phases, $\mathcal{P}_n = \{I, X, Y, Z\}^{\otimes n} \times \{\pm 1, \pm i\}$. Therefore, any d-dimensional unitary operators C, where $d = 2^n$, is an element of $\mathcal{C}_n$ if and only if their conjugation satisfies $CPC^\dagger \in \mathcal{P}_n\ \forall\ P \in \mathcal{P}_n$. Note that the Pauli group actually comprises a unitary 1-design [69]. The single qubit Clifford group, where $n = 1$, can be generated by the phase gate and Hadamard gate defined in Eq.(2.18). For higher qubits, $\mathcal{C}_n$ generation requires an additional CNOT gate and in general tensor products of three different gates. For universal quantum computation, non-Clifford gates are needed, along with a procedure known as magic state distillation with the Clifford group [74, 75]. Common examples of non-Clifford gates are the T-gate and the Toffoli gate.

2.3.5 Non-Clifford 2-Designs of the Project

The gate sets that will be investigated by the RB protocol are presented here[3]. These are a series of t-designs, conjectured by Dilkes [38] which were built upon previous work of the supervisor. No derivations will be given, hence only the mathematical method of generating the gates are presented. The targeted non-Clifford gate set $\mathcal{J}_{4n}$ of size $4n$ where $\mathcal{J}_{4n} = \{U_1, U_2, \ldots, U_{4n}, \forall\ n \geq 3\}$, with unitary gates $U_i \in \mathcal{J}_{4n}$, can be constructed by

$$\mathcal{J}_{4n} = \{\mathcal{P} \times B \times \{I, C_n, C_n^2, \ldots, C_n^{n-1}\} \times B^\dagger\}. \tag{2.57}$$

The generating matrices C_n and B are given by

$$C_n = \begin{pmatrix} e^{\frac{2\pi i}{n}} & 0 \\ 0 & e^{-\frac{2\pi i}{n}} \end{pmatrix}, \tag{2.58}$$

$$B = \frac{1}{\sqrt{6 - 2\sqrt{3}}} \begin{pmatrix} 1 - i & \sqrt{3} - 1 \\ \sqrt{3} - 1 & -1 - i \end{pmatrix}, \tag{2.59}$$

and $\mathcal{P} = \{I, X, Y, Z\}$ is once again the Pauli group containing the identity I and the Pauli matrices $\{X, Y, Z\}$. Note that here only $n \geq 3$ will be concerned since that is the condition for forming a unitary 2-design, i.e. it satisfies Eq.(2.47). Crucially for $n > 3$, these 2-designs do not satisfy completely the requirement of Eq.(2.56), thereby rendering them non-Clifford. In the simplest case where $n = 3$, $\mathcal{J}_{4n}$ simply reduces down to a total of 12 gates, i.e.

$$\mathcal{J}_{12} = \{\{I, X, Y, Z\} \times B \times \{I, C_3, C_3^2\} \times B^\dagger\} \tag{2.60}$$

which does form the Clifford group.

2.4 Fidelity

2.4.1 Fidelity of a State

The fidelity in terms of quantum states is simply a measure of distance or closeness between quantum states. The definition of fidelity for two arbitrary mixed states ρ and σ is simply given by [43]

$$F(\rho, \sigma) = \left(\mathrm{tr}\sqrt{\sqrt{\rho}\sigma\sqrt{\rho}}\right)^2. \tag{2.61}$$

[3]Heavily adapted from interim report "Single Qubit Randomised Benchmarking of Non-Clifford Gates"

Here some properties of the fidelity useful to the project are presented. It has a value bounded by $0 \leq F(\rho, \sigma) \leq 1$, with the fidelity only equal to 1 when the states are identical i.e. $\rho = \sigma$. The fidelity is symmetric in its arguments, such that $F(\rho, \sigma) = F(\sigma, \rho)$ for all ρ and σ. Suppose that a pure state goes under some quantum operation and outputs a noisy state. In such a case σ is the pure state $|\psi\rangle\langle\psi|$ and ρ represents the noisy mixed state, then their fidelity simply reduces down to $F(|\psi\rangle\langle\psi|, \rho) = \langle\psi|\rho|\psi\rangle$. Finally, the fidelity is invariant under all unitary transformations, where $F(U\rho U^\dagger, U\sigma U^\dagger) = F(\rho, \sigma)$.

2.4.2 Fidelity of a Channel

Similar to that of quantum states, the fidelity in terms of quantum channels measures the closeness between them but with a few variations. The fidelity of a channel typically draws the closeness comparison between a quantum channel and an identity channel. Hence for an arbitrary CPTP map $\mathcal{E}$ which describes a quantum operation, one usually talks about the *average gate fidelity* [76]

$$\begin{aligned} \bar{F}(\mathcal{E}, I) &= \bar{F}(\mathcal{E}) \\ &= \int d\psi \ \langle\psi|\mathcal{E}(\psi)|\psi\rangle \\ &= \int d\psi \ \mathrm{tr}(|\psi\rangle\langle\psi|\mathcal{E}(\psi)). \end{aligned} \tag{2.62}$$

This can be thought of as the overlap between a pure state ψ and its processed state averaged over all pure states. This is because the integral is computed with respect to the Haar measure, which is a uniform distribution over all pure states ψ, with a normalisation $\int d\psi = 1$. An extension of this definition can be made to how closely $\mathcal{E}$ can approximate a unitary transformation $\mathcal{U}$,

$$\begin{aligned} \bar{F}(\mathcal{E}, \mathcal{U}) &= \int d\psi \ \langle\psi|U^\dagger\mathcal{E}(\psi)U|\psi\rangle \\ &= \int d\psi \ \mathrm{tr}(\mathcal{U}(\psi)\mathcal{E}(\psi)), \end{aligned} \tag{2.63}$$

this time with the overlap between all unitarily transformed pure states $\mathcal{U}(\psi)$ and all operated pure states $\mathcal{E}(\psi)$.

In section 2.3.2, it was presented that the 2-design twirl on an arbitrary Pauli-Liouville channel would result in the depolarising channel in Pauli-Liouville representation with the form

$$\boldsymbol{\mathcal{D}}_p = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & p & 0 & 0 \\ 0 & 0 & p & 0 \\ 0 & 0 & 0 & p \end{pmatrix}. \tag{2.64}$$

Since the noise maps used are TP, this implies that the first diagonal value is equal to one as it was shown in subsection 2.2.3 that this is the condition for an arbitrary TP map in this representation. This is equivalent to the depolarising channel in the usual form

$$\mathcal{D}_p(\rho) = (1-p)\rho + p\frac{I}{2}. \tag{2.65}$$

Since this is now a diagonal matrix, it implies that the sequential composition of many of these noise map will only result in the power up of the diagonal elements. In both cases, the repeated application of the depolarising channels transforms a pure state into a maximally mixed state by a factor of p. Finally, the average gate fidelity of the channel relating to the depolarising parameter has a simple known form [77]

$$\bar{F}(\boldsymbol{\mathcal{D}}_p) = \frac{p(d-1)+1}{d}, \tag{2.66}$$

where d is the dimension of the Hilbert space. The RB protocol allows p to be extracted and Eq.(2.66) to be calculated. This in turn is an estimation on the fidelity of the original noise map $\boldsymbol{\Lambda}$, where its exact average gate fidelity with respect to the identity channel in Pauli-Liouville representation is [78]

$$\bar{F}(\boldsymbol{\Lambda}) = \frac{\mathrm{tr}(\boldsymbol{\Lambda}) + d}{d(d+1)} \approx \bar{F}(\boldsymbol{\mathcal{D}}_p). \tag{2.67}$$

2.5 Randomised Benchmarking

Fundamentally, a noise channel undergoing the randomised benchmarking (RB) protocol gets transformed into a depolarising channel, ending up with average gate fidelity equal to that of a depolarising channel. Since the expression of the average gate fidelity related to the channel's depolarising parameter is known and also practically measurable, this in turn allows for the measurement of the decay rate and hence the fidelity of the original noise channel. The noise model of this project are assumed to be time-independent and gate-independent, which means the noise channel applied is assumed to have no dependence on the gate applied. The noise model is also assumed to be Markovian, which means it does not depend on any gates that were previously applied in the sequence.

When it comes to how successful an initially prepared state is measured, one usually talks about the *survival probability* of the measurement. That is, how much of the original state has "survived" after a series of computation. Therefore, if a state is prepared and measured with no gates in between, the success of measurement would solely be affected by the SPAM errors. If multiple depolarising channels are applied between state preparation and measurement, the increase of failure to measure a state is solely affected by the channel's depolarising parameter as the state is becoming more and more depolarised. Therefore, by recording the how the survival probability decreases with the amount of depolarising channels applied, the depolarising parameter can be extracted out of the decay curve of the survival probability independent of SPAM errors. Ultimately, the average fidelity of the noise channel is then obtained. A circuit diagram of the RB protocol is shown in Fig. 2.2.

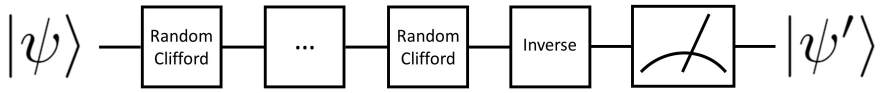

Figure 2.2: The circuit diagram for the randomised benchmarking procedure.

2.5.1 Zeroth Order Model RB

The standard model of RB is first reviewed in the Kraus representation[4]:

1. Draw a random sequence of gates $\{U_r\}_{r=1,\dots,s}$ with sequence length s independently and uniformly at random, from a unitary gate set $\mathbf{G}$ which forms a 2-design.

2. Obtain K_s sequences of sequence length s by repeating step 1 K_s times. $U_j^i \in \mathbf{G}$ is denoted as the jth unitary gate of the ith sequence, such that $1 \leq i \leq K_s$ and $1 \leq j \leq s$. Compute an inverse element for all sequences as $U_{s+1}^i := (U_s^i U_{s-1}^i \dots U_1^i)^\dagger$. If U_j^i is a single qubit gate or if $\mathbf{G}$ forms a group such as the Clifford group, then this can be done efficiently [71].

3. Prepare an input qubit $\rho := |\psi\rangle\langle\psi|$. This is the ideal state to its noisy preparation $\tilde{\rho}$.

4. Apply the sampled gates of the ith sequence from step 1 and the sequence reversal gate U_{s+1}^i in the end. The noisy implementation of the unitary transformation by Eq.(2.21) is described by the aforementioned channel composition

$$\tilde{\mathcal{U}}_j^i(\rho) = \mathcal{D}_j^i \circ \mathcal{U}_j^i(\rho). \tag{2.68}$$

   $\mathcal{D}_j^i$ represents a noise channel in the form of an arbitrary CPTP map. The noisy evolution of the initially prepared noisy (mixed) state $\tilde{\rho}$ under a particular ith sequence is

$$\tilde{\mathcal{U}}^i(\tilde{\rho}) := \bigcirc_{j=1}^{s+1} \tilde{\mathcal{U}}_j^i(\tilde{\rho}) = \bigcirc_{j=1}^{s+1} [\mathcal{D}_j^i \circ \mathcal{U}_j^i](\tilde{\rho}) \tag{2.69}$$

5. At the end of each sequence, make a POVM measurement with $\{\tilde{E}_\rho, I - \tilde{E}_\rho\}$, which is the

[4]Heavily adapted from interim report "Single Qubit Randomised Benchmarking of Non-Clifford Gates" [1].

noisy version of ideal POVM measurement $\{\rho, I-\rho\}$. This gives the ith *survival probability*

$$\mathrm{tr}\Big[\tilde{E}\tilde{\mathcal{U}}^i(\tilde{\rho})\Big]. \tag{2.70}$$

6. Average over all survival probabilities for all K_s random sequences $\tilde{U}_i$ gives the *sequence fidelity*

$$F_{\mathbf{G}}(s, K_s) := \frac{1}{K_s}\sum_{i=1}^{K_s}\mathrm{tr}\Big[\tilde{E}\tilde{\mathcal{U}}^i(\tilde{\rho})\Big]. \tag{2.71}$$

In the limit of large number of sequences K_s, $F_{\mathbf{G}}(s, K_s)$ asymptotically converges rapidly to $F_{\mathbf{G}}(s)$ as

$$F_{\mathbf{G}}(s) = \frac{1}{|\mathbf{G}|^s}\sum_{i=1}^{|\mathbf{G}|^s}\mathrm{tr}\Big[\tilde{E}\tilde{\mathcal{U}}^i(\tilde{\rho})\Big] \tag{2.72}$$

$F_{\mathbf{G}}(s)$ is now a uniform average over all $|\mathbf{G}|^s$ possible sequences. This asymptotic property makes the RB an efficient and scalable protocol for more qubits and longer sequence length s [28, 79].

By the original derivation in Ref.[28], the estimation of $F_{\mathbf{G}}(s)$ over increasing sequence length s will manifest itself as an exponential decay curve, with a decay equation of

$$F_{\mathbf{G}}(s) \approx Ap^s + B, \tag{2.73}$$

where p is the depolarising parameter related to the average gate fidelity $\bar{F}_{\mathbf{G}}$ by $\bar{F}_{\mathbf{G}} = (p+1)/2$. The line fit of Eq.(2.73) then approximates the average gate fidelity. Providing that the noises are gate-independent and time-independent, RB differentiates SPAM errors and isolate them into parameters A and B, effectively making them "nuisance". This is known as the "zeroth model" of the RB protocol and an example of a typical decay curve is shown in Fig. 2.3.

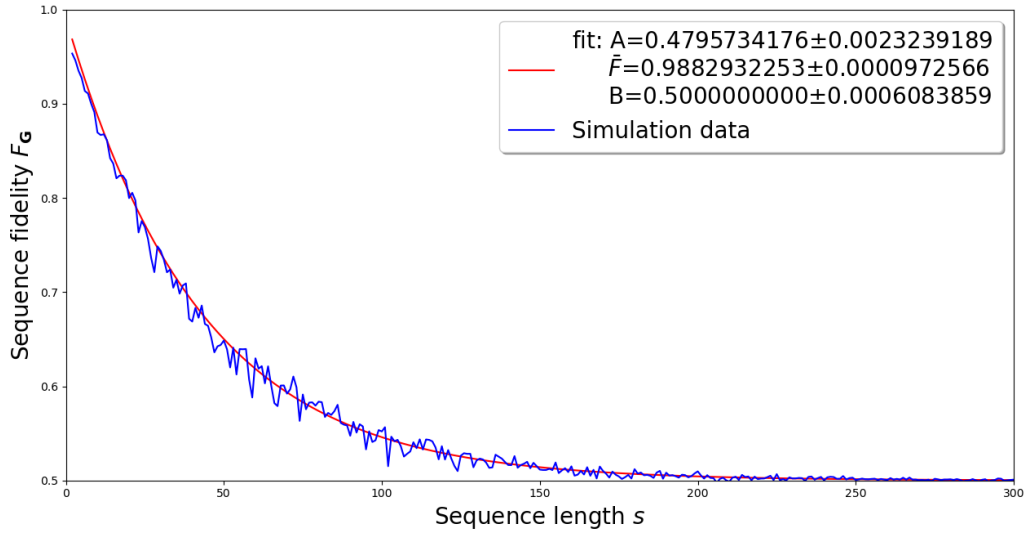

Figure 2.3: A typical zeroth model RB decay curve where the sequence fidelity is plotted against sequence length. Here the survival probability is computed for every sequence length up to $s = 300$. The curve is directed fitted to $A(2\bar{F}-1)^s + B$ instead to directly extract the average gate fidelity $\bar{F}$, which is 0.9883 (to 4 s.f.) in this case. Notice here the sequence fidelity has a floor of 0.5 which is the offset of the curve.

2.5.2 RB with Constant Elimination

In this version of RB, two essential modifications are made to the standard RB protocol as laid out in section (2.5.1), adapted from *Helsen et al.*[80]. The protocol will be described in Pauli-Liouville representation and will be used as the main RB protocol for all simulations of this project.

The first adjustment is to prepare two different input states ρ_1 and ρ_2, perform on them the same randomised benchmarking sequence and then subtract their survival probabilities. The second modification is to perform the measurement with a pre-determined Pauli operator $\mathcal{P}_i$. The results should be consistent for any option of Pauli matrices but a conventional choice tends to be $\mathcal{P}_i = Z$ for experiments. In addition, the input states ρ_1 and ρ_2 are initially prepared to be $(I + \mathcal{P}_i)/2$ and $(I - \mathcal{P}_i)/2$ respectively. Under the choice of $\mathcal{P}_i = Z$, the prepared states are just the up and down states $\rho_\uparrow$, $\rho_\downarrow$ with $\rho_\uparrow = |\uparrow\rangle\langle\uparrow| = |0\rangle\langle 0|$ and $\rho_\downarrow = |\downarrow\rangle\langle\downarrow| = |1\rangle\langle 1|$ respectively.

The protocol is laid out as follows:

1. This is the same as step 1 and 2 of the zeroth order model. The Pauli-Liouville form of the unitary channels are denoted as $\boldsymbol{\mathcal{U}}_r$ with matrix entries $\mathrm{tr}(\mathcal{P}_i U_r \mathcal{P}_j U_r^\dagger)$ and $\mathcal{P}_i, \mathcal{P}_j \in \mathcal{P}/\sqrt{2}$. $\boldsymbol{\mathcal{U}}_j^i$ is now the jth unitary channel of the ith sequence, with $1 \leq i \leq K_s$ and $1 \leq j \leq s$. Compute the inverse channel $\boldsymbol{\mathcal{U}}_{s+1}^i = (\boldsymbol{\mathcal{U}}_s^i \boldsymbol{\mathcal{U}}_{s-1}^i ... \boldsymbol{\mathcal{U}}_1^i)^\dagger$.

2. Prepare two input qubits $\rho_1 = (I + \mathcal{P}_i)/2$ and $\rho_2 = (I - \mathcal{P}_i)/2$. Here $\mathcal{P}_i$ is chosen to be Z and hence the subsequent up and down states are denoted as $|\rho_\uparrow\rangle\rangle$ and $|\rho_\downarrow\rangle\rangle$ respectively in their Pauli-Liouville form. Their noisy preparation are therefore $|\tilde{\rho}_\uparrow\rangle\rangle$ and $|\tilde{\rho}_\downarrow\rangle\rangle$.

3. For the same ith sequence, apply the sampled gates followed by the sequence reversal channel $\boldsymbol{\mathcal{U}}_{s+1}^i$ on both qubits separately, i.e.

$$\begin{aligned} |\tilde{\rho}_\uparrow\rangle\rangle &\to \tilde{\boldsymbol{\mathcal{U}}}_{s+1}^i \tilde{\boldsymbol{\mathcal{U}}}_s^i ... \tilde{\boldsymbol{\mathcal{U}}}_1^i |\tilde{\rho}_\uparrow\rangle\rangle \\ |\tilde{\rho}_\downarrow\rangle\rangle &\to \tilde{\boldsymbol{\mathcal{U}}}_{s+1}^i \tilde{\boldsymbol{\mathcal{U}}}_s^i ... \tilde{\boldsymbol{\mathcal{U}}}_1^i |\tilde{\rho}_\downarrow\rangle\rangle, \end{aligned} \tag{2.74}$$

   where $\tilde{\boldsymbol{\mathcal{U}}}_j^i = \boldsymbol{\mathcal{D}}_j^i \boldsymbol{\mathcal{U}}_j^i$ which is the noisy implementation of the ideal unitary channel and $\boldsymbol{\mathcal{D}}_j^i$ is a noise channel (CPTP map) in the form of Eq.(2.38).

4. Perform a POVM measurement with $\{Z, I - Z\}$ at the end of each sequence for both qubits. Their survival probabilities are simply

$$\begin{aligned} p_\uparrow^i(\tilde{\rho}_\uparrow) &= \langle\langle Z | \tilde{\boldsymbol{\mathcal{U}}}_{s+1}^i \tilde{\boldsymbol{\mathcal{U}}}_s^i ... \tilde{\boldsymbol{\mathcal{U}}}_1^i | \tilde{\rho}_\uparrow \rangle\rangle \\ p_\downarrow^i(\tilde{\rho}_\downarrow) &= \langle\langle Z | \tilde{\boldsymbol{\mathcal{U}}}_{s+1}^i \tilde{\boldsymbol{\mathcal{U}}}_s^i ... \tilde{\boldsymbol{\mathcal{U}}}_1^i | \tilde{\rho}_\downarrow \rangle\rangle. \end{aligned} \tag{2.75}$$

   Compute half of their difference $(p_\uparrow^i(\tilde{\rho}_\uparrow) - p_\downarrow^i(\tilde{\rho}_\downarrow))/2$.

5. Repeat step 2-4 for all of the K_s sequences to estimate the sequence fidelity

$$F_{\mathbf{G}}(s, K_s) = \frac{1}{K_s} \sum_{i=1}^{K_s} \frac{1}{2} (p_\uparrow^i(\tilde{\rho}_\uparrow) - p_\downarrow^i(\tilde{\rho}_\downarrow)) \tag{2.76}$$

The survival probability is again estimated for an increasing sequence length s and fitted, but this time to a simpler exponential curve of

$$F_{\mathbf{G}}(s) \approx A p^s, \tag{2.77}$$

and an example of the decay curve is shown in Fig. 2.4. The fitting parameter B from before is now removed due to the first modification, since it changes the exponential fit with non-zero offset that is intrinsic to the RB to one with zero offset. In addition, the first adjustment also has the benefit of reducing RB's statistical fluctuations due to the cancelling of both offsets from the up and down states, which is perceptible when the sequence length increases [79]. The second adjustment to the zeroth order RB protocol was argued to further improve the accuracy of fidelity estimates since it also contributed independently to the

elimination of the offset. Although it may seem like the proposed measurement of identical sequence on two different input states may double the overhead in the required sample numbers per sequence, *Helsen et al.* argued that performing RB with different input states and the specific measurement operator as described previously is not anymore costly than the standard RB. For the exact rigorous arguments, the readers are referred to the adapted paper Ref.[80].

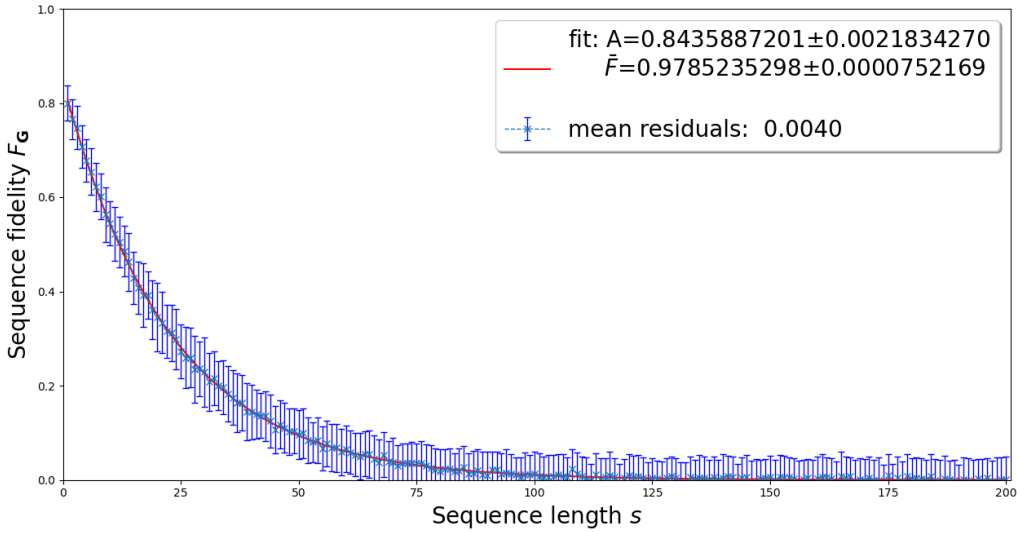

Figure 2.4: A typical RB decay curve where one of the SPAM constant is eliminated. The sequence fidelity is plotted against sequence length. Here the survival probability is computed for every sequence length up to $s = 200$. The curve is directed fitted to $A(2\bar{F} - 1)^s$ instead to directly extract the average gate fidelity $\bar{F}$, which is 0.9785 (to 4 s.f.) in this case. Notice here the floor of the sequence fidelity is zero, due to the offset of the curve being removed.

Chapter 3

Results

3.1 Numerical Simulations

To be concise with the terminology, the value $\bar{F}$ is the average gate fidelity, and will sometimes be shortened to just fidelity. This is different to sequence fidelity, which is used in text without shortening. Firstly, the methodology of generating the noise maps recommended by Dr Robin Harper for the RB simulation are explained. The noise channel is generated as

$$\boldsymbol{\mathcal{E}} = \boldsymbol{\mathcal{U}}\boldsymbol{\Lambda}\boldsymbol{\mathcal{V}} \tag{3.1}$$

which is a single qubit random arbitrary CPTP map in the Pauli-Liouville representation. For the map to first be TP, the first diagonal element of $\boldsymbol{\Lambda}$ is set to 1 with the form of

$$\boldsymbol{\Lambda} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ t_1 & \lambda_1 & 0 & 0 \\ t_2 & 0 & \lambda_2 & 0 \\ t_3 & 0 & 0 & \lambda_3 \end{pmatrix}. \tag{3.2}$$

Since it was shown that the random noise map will be transformed into a depolarising channel, it implies the rest of the diagonal elements are closely linked to the average gate fidelity of the simulation. The idea is to draw these random channels to be close to the Identity channel. Therefore, for high fidelity the diagonal elements λ_1, λ_2 and λ_3 should be generated arbitrarily such that their average is close to unity. Note that if all diagonal elements are equal to one, then the noise map is simply trivial as it would just be the identity matrix i.e. absence of noise. To do this a normal distribution with a high mean (with respect to the accepted interval) is used so that the λs can be generated continuously and only accepted if it is within an interval of high fidelity that is desired. For this project, the accepted interval for high fidelity is set to be $[0.9, 1]$. This in effect truncates the Gaussian distribution that is being used. As for the mean and standard deviation of the Gaussian distribution, they are actually dependent on the type of quantum computing architecture in question, and are therefore arbitrary. Under the recommendation of Dr. Harper, the mean and standard deviation for the state preparation, control and measurement are chosen to be (0.985, 0.15), (0.998, 0.04) and (0.98, 0.15) respectively. His motivation of choice of these numbers stemmed from looking at IBM's devices that has a superconducting architecture. By comparing their operational gates and state preparation and measurement of random channels, it seems that the SPAM errors tend to dominate compared to the single qubit gate errors, and it is a typical consequence of superconducting hardware. Now that three λs are generated, for a TP map to also be CP, the λs have to satisfy the aforementioned constraints $|1 \pm \lambda_3| \geq |\lambda_1 \pm \lambda_2|$. Hence for a particular λ_1 and λ_2, a generated λ_3 is discarded until this constraint is met, in order for the generated map to be CPTP. It can be seen on Fig. 3.1a all of the required constraints above led to noise channels with a slightly asymmetrical normal distribution of fidelity ranging between 0.94 to 1. For the purpose for this project, the relevant noise map will be chosen to be unital and so the non-unital part is set to zero. Finally, the rotation matrices $\boldsymbol{\mathcal{U}}$ and

$\boldsymbol{\mathcal{V}}$, previously presented as

$$\mathcal{U}, \boldsymbol{\mathcal{V}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta_1 & \sin\theta_1 & 0 \\ 0 & -\sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \cos\theta_2 & \sin\theta_2 \\ 0 & 0 & -\sin\theta_2 & \cos\theta_2 \end{pmatrix} * \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta_3 & \sin\theta_3 & 0 \\ 0 & -\sin\theta_3 & \cos\theta_3 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{3.3}$$

need to be generated for small angles. This is also done by generating θ_1, θ_2 and θ_3 using a normal distribution with a mean of zero and a standard deviation of 1 and then multiplied by a small factor of r. For state control and SPAM, they are chosen to be 0.05 and 0.06 respectively. The spread of 10^7 random noise maps generation is shown in Fig. 3.1.

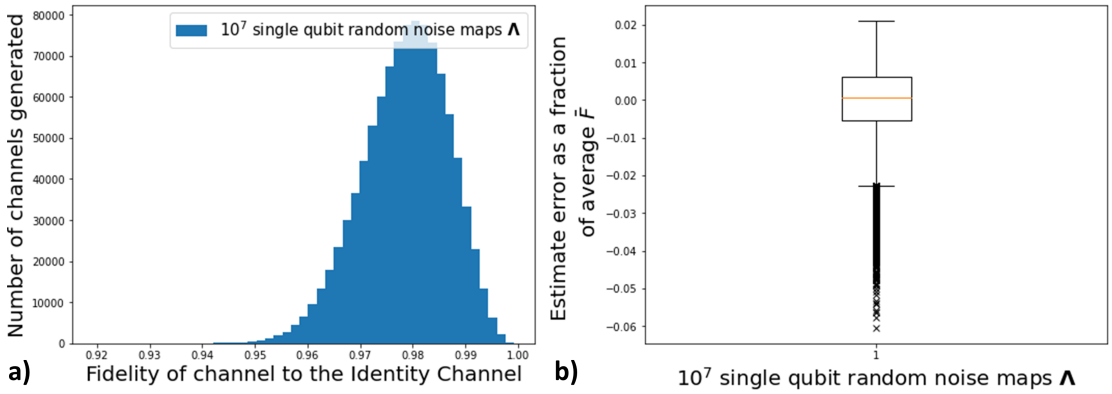

Figure 3.1: **a)** The spread of the fidelity (Eq.(2.67)) of 10^7 random noise maps generated to be near Identity, plotted as a histogram. **b)** By taking the average of $\bar{F}_i$ of all 10^7 random noise maps as the estimated true $\bar{F}_{\text{true}}$, the estimated error $(\bar{F}_i - \bar{F}_{\text{true}})/\bar{F}_{\text{true}}$ are plotted in a Tukey/Box plot, with $\bar{F}_{\text{true}} = 0.9785195$.

Before the simulation is performed, there is a way to check if the algorithm written is performing correctly, such that the simulation agrees with the theory. Instead of running the RB protocol with noisy sequences, an initial noiseless test run can be performed for arbitrary sequence lengths. In theory, the simulation should return a perfect survival probability for any sequence length in the absence of SPAM and state control noise. This is because within the calculation, the sequence of gates applied followed by a reversal gate would not have accumulated any noise at all, allowing the prepared state to survive completely. This is shown in Fig. 3.2. Note that for a particular sequence length s, the sequence fidelity is calculated with the average of survival probabilities of different sequences of the same length s. The survival probabilities for all these sequences are one, since the sequences are noiseless, resulting to also a sequence fidelity of one for every sequence length, which agrees with the theory. Similarly, if only SPAM errors were present, then the sequence fidelity should have a roughly constant value, without the exponential decay curve, which agrees with Fig. 3.3.

Preparation (mean)	**Measurement (mean)**	**Sequence fidelity reduction (%)**
0.9950	0.9900	11.92
0.9850	0.9800	12.00
0.9750	0.9700	12.06
0.9650	0.9600	12.09
0.9550	0.9500	12.09

Table 3.1: The mean for both Gaussian distributions used to generate the state preparation and measurement errors and their corresponding reduction in sequence fidelity. Their standard deviations are both 0.15.

The curve fit of the simulation is done by using Numpy's internal curve fit function, which uses the standard non-linear least squares analysis. The Python code is adapted with the `MPI4PY` module, which

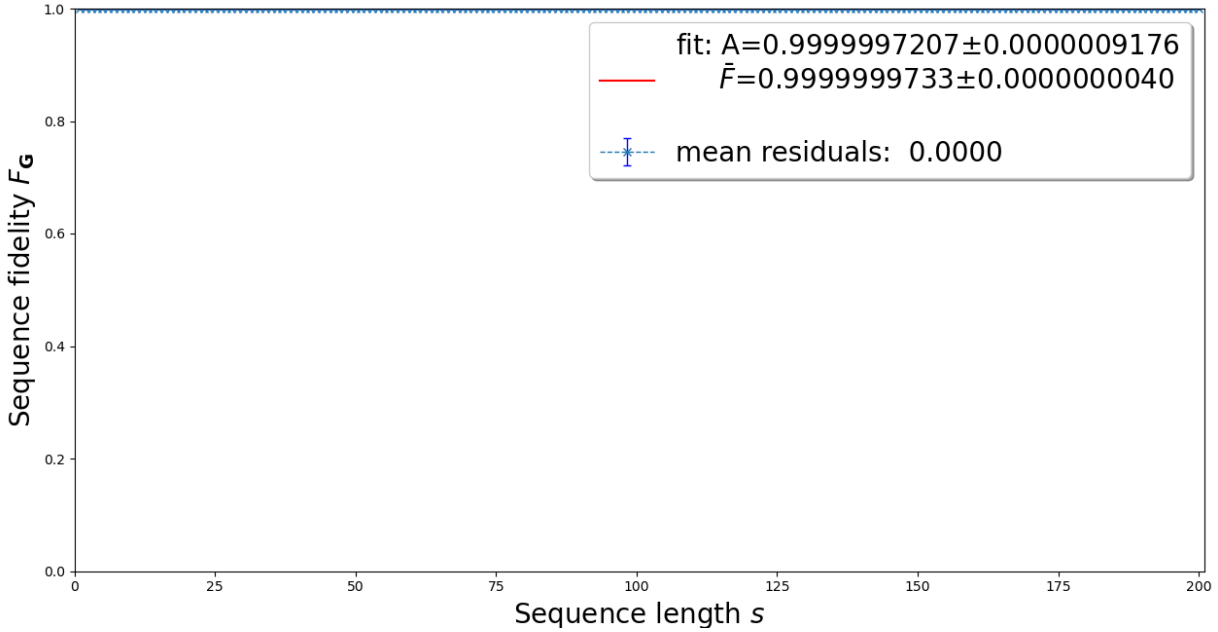

Figure 3.2: The RB curve for a noiseless experiment with perfect sequence fidelities (modulo computational floating point error).

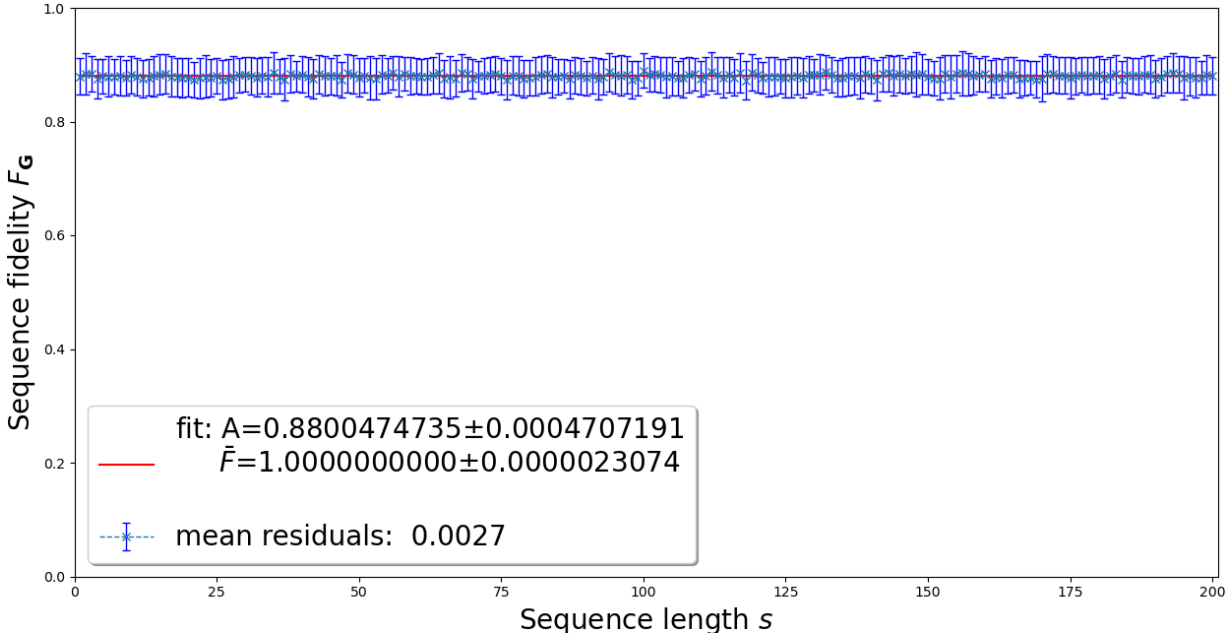

Figure 3.3: The RB curve for experiment where applied sequences are noiseless, however with SPAM errors present with their aforementioned parameters. Mean and standard deviation of (0.998, 0.04) and (0.98, 0.15) for state preparation and measurement respectively. This was ran for $K_s = 100$ up to $s = 200$ with the Clifford group t-designs $n = 3$. Error bars are standard deviation of the mean and the mean residuals are irrelevant here.

allows for the parallelisation of computation which is then ran on the University of Bristol's supercomputer BlueCrystalp3 using 48 CPU cores. In order to investigate the infinite family of gate sets $\mathcal{J}_{4n}$ where $\mathcal{J}_{4n} = \{U_1, U_2, ..., U_4, \forall n \geq 3\}$, the RB protocol is first ran for $n = 3$ which does form a Clifford group. From here onwards, the number of sequences K_s that give K_s different survival probabilities, which are then averaged over to give the sequence fidelity for a particular sequence length s, will be simply called

sequence samples. Since Clifford gates which form a t-design are standard in RB, this allows for the basic verification of the theory as well as investigation of how the sequence length s and sequence samples K_s affects the curve. Here the sequence sample is chosen to be $s = 200$ and RB protocol is ran for $K_s = 5, 10, 50, 100, 500, 1000$. In order to investigate the estimate of the average gate fidelity of the noise map $\bar{F}(\boldsymbol{\mathcal{D}}_p)$, the estimated average gate fidelity averaged over many different RB curves are compared with the (estimated) exact average gate fidelity $\bar{F}(\boldsymbol{\mathcal{E}})$ averaged over 10^7 randomly generated noise maps $\boldsymbol{\mathcal{E}}$, for both Clifford $n = 3$ and non-Clifford $n = 50$ gates. They are compared against increasing resource number, which is simply the sequence length s multiplied by the sequence samples K_s. Again $s = 200$ and the values $K_s = 5, 10, 50, 100, 500, 1000$ are chosen to match previous investigation of increasing sequence samples. Since the estimates are expected to converge rapidly, it would not be meaningful to plot resource numbers with equal increments. They are shown in Fig. 3.7 and Fig. 3.9 respectively.

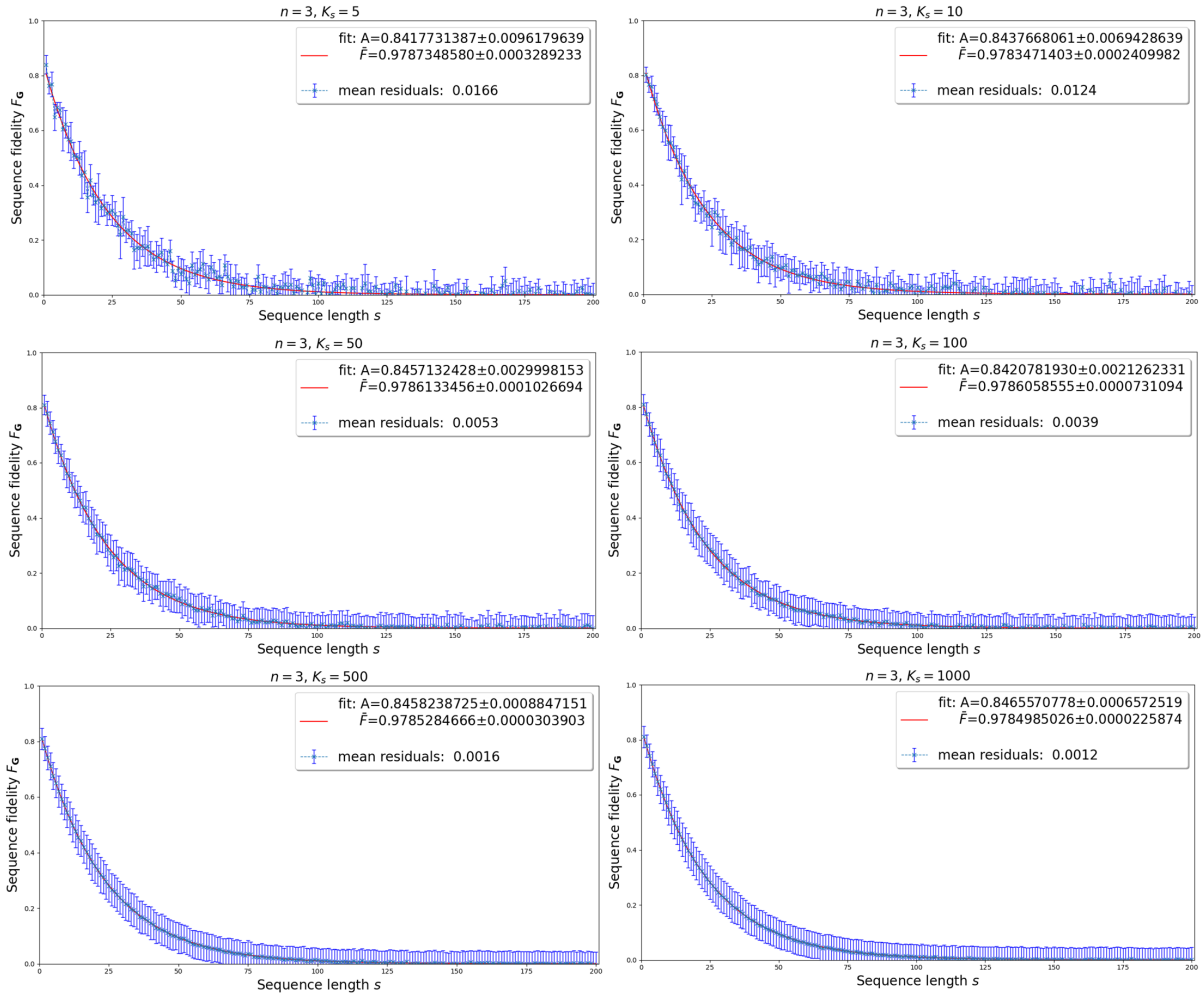

Figure 3.4: The RB curve produced for t-designs Clifford gate set $n = 3$ with increasing sample sizes. Here the error bars are the standard deviation of the mean. The mean residuals are the average of all the distances between the fitted values and the simulated values. This also shows that the variance is independent of the sequence samples.

3.2 Discussion

The overall discussion of the results section and their implications are given here.

It has been confirmed that the numerical foundation of the RB protocol has been implemented correctly. In the simplest non-idealised and non-physical case where the system is not affected by any noise

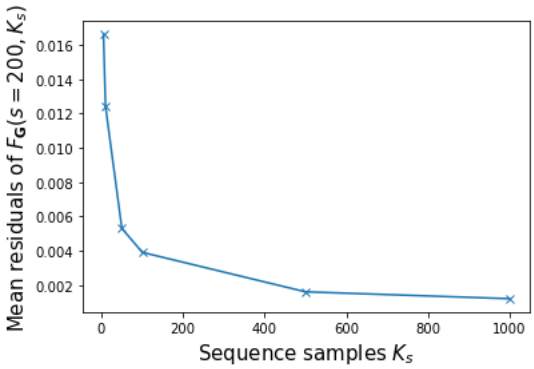

Figure 3.5: The mean residuals of the sequence fidelity plotted against the sequence samples K_s according to Fig. 3.4.

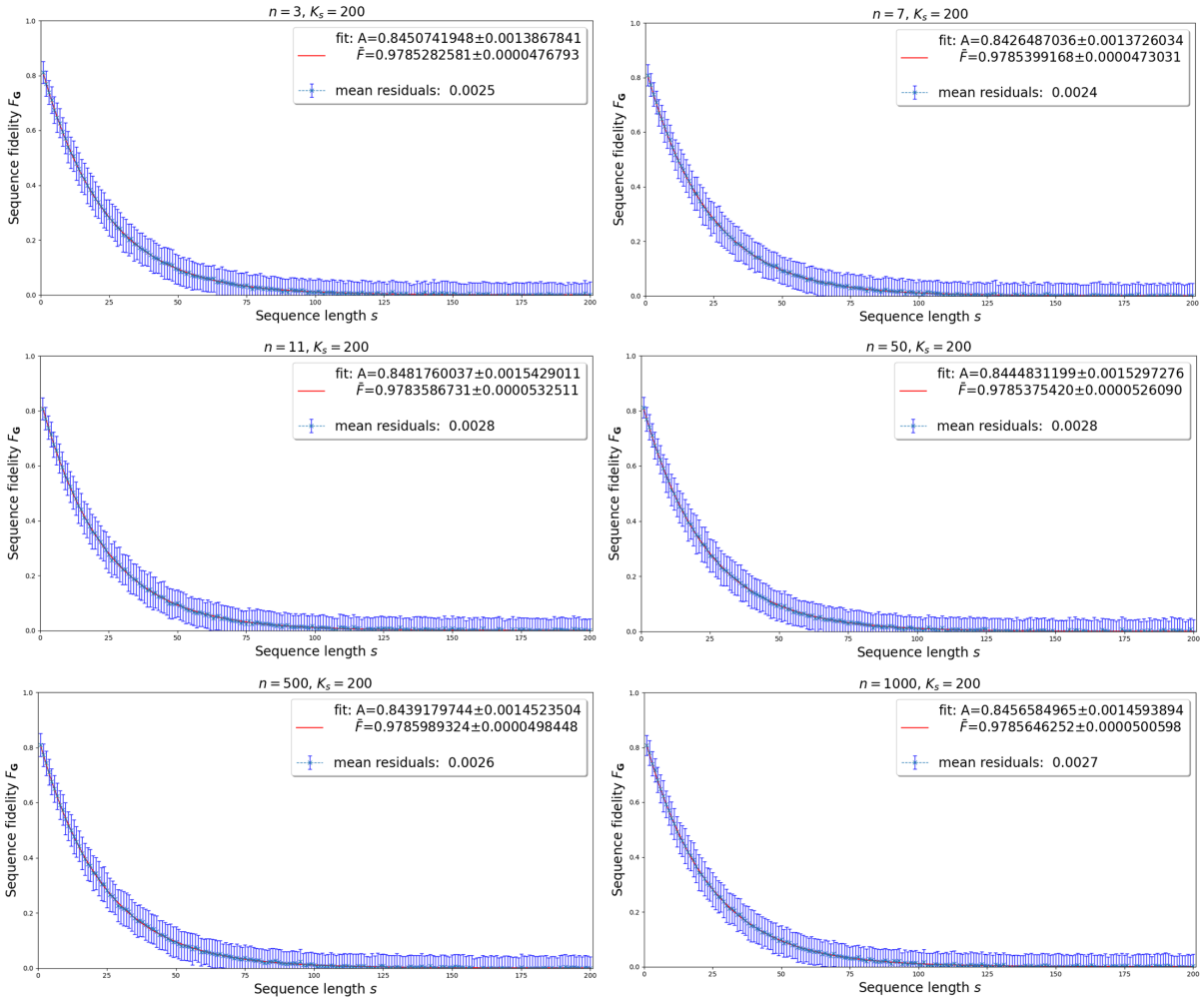

Figure 3.6: The RB curve produced for t-designs non-Clifford gate sets of $n = 3, 7, 11, 50, 500, 1000$ with fixed sample size of $K_s = 200$ for every sequence length s. Here the error bars are the standard deviation of the mean. The mean residuals are the average of all the distances between the fitted values and the simulated values.

whatsoever, the sequence fidelity for every sequence s is exactly one, which corresponds to a maximum overlap between the measurement and the state and thus the state survived perfectly. In a more physical

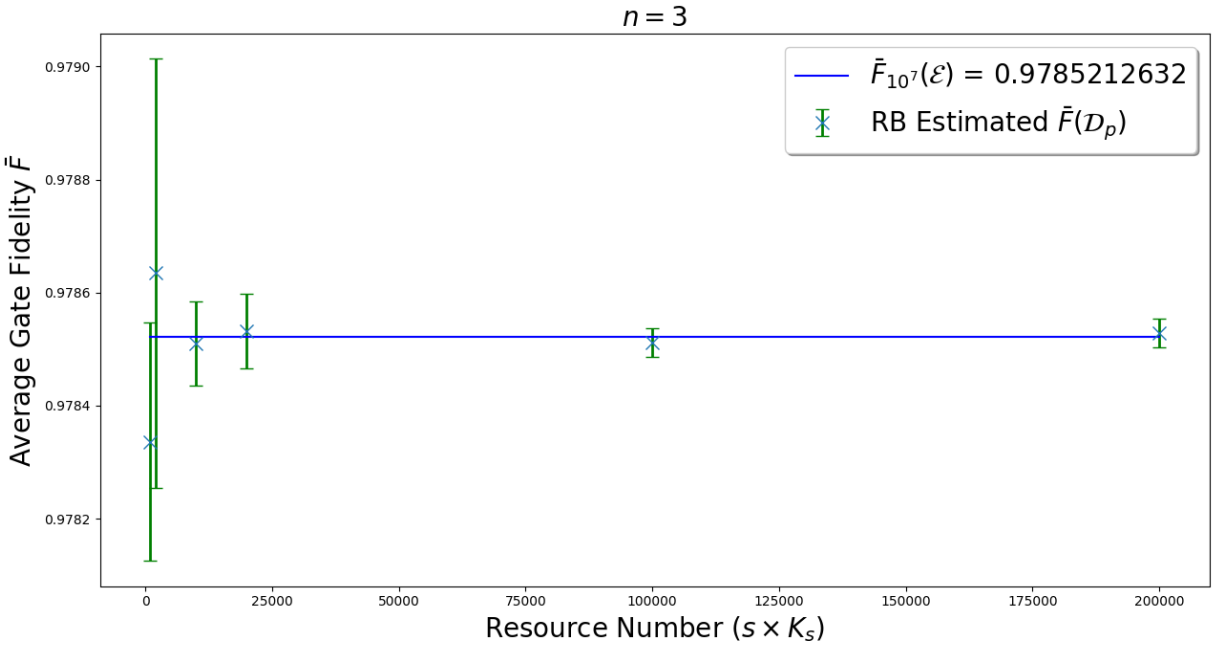

Figure 3.7: The averaged gate fidelity $\bar{F}$ estimates with increasing resource number $200 \times \{5, 10, 50, 100, 500, 1000\}$ for Clifford gates $n = 3$. Here for each resource number, $\bar{F}$ is averaged over 10 independent estimates of $\bar{F}$ from the RB protocol.

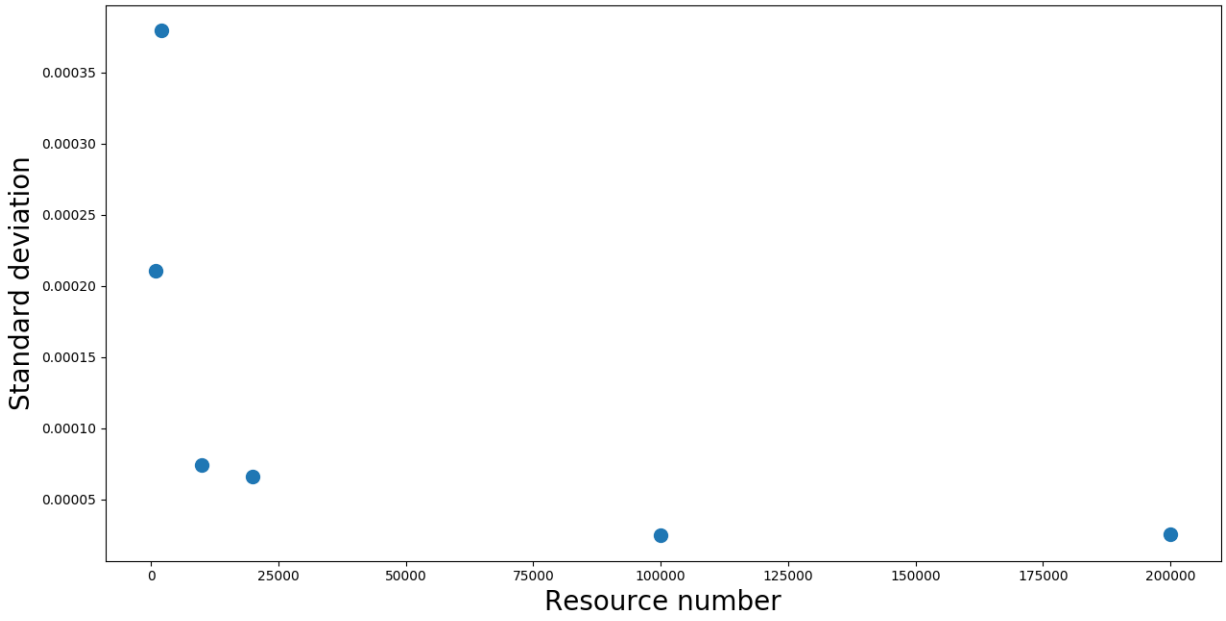

Figure 3.8: The standard deviation of the 10 estimated data points for each resource number for Fig. 3.7.

scenario, the quantum state is noisily prepared in a lab and measured right away with a noisy measurement, with no gates applied in between. Under the influence of (superconducting architectures based) SPAM errors, Fig. 3.3 showed that the sequence fidelities fluctuate slightly around the fitted parameter $A = 0.88004 \pm 0.0005$, implying that the SPAM errors alone have a significant impact of reducing the sequence fidelities by 12.0% for all sequence length s in this case. While one might think that the exact value sequence fidelity reduction is heavily dependent on the parameters of the Gaussian distribution that were used to generate the diagonal elements of the noise maps, table 3.1 shows that this is actually not the case. The SPAM errors cannot actually be reduced by making the mean of the Gaussian distributions

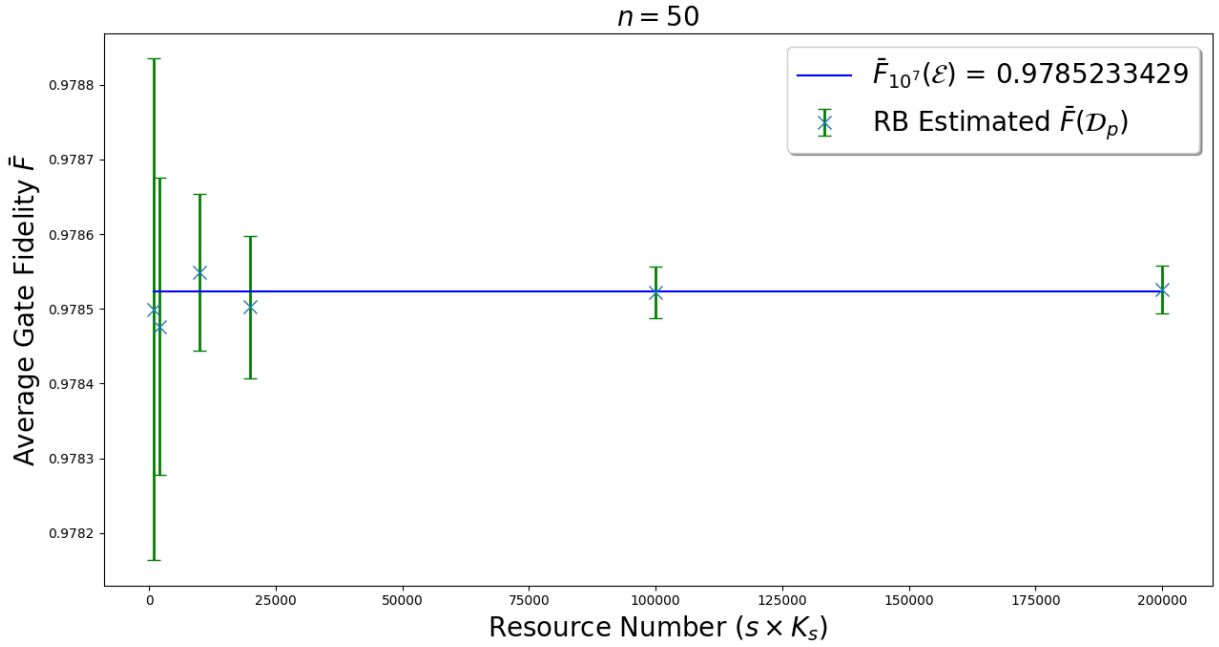

Figure 3.9: The averaged gate fidelity $\bar{F}$ estimates with increasing resource number $200 \times \{5, 10, 50, 100, 500, 1000\}$ for Clifford gates $n = 50$. Here for each resource number, $\bar{F}$ is averaged over 10 independent estimates of $\bar{F}$ from the RB protocol.

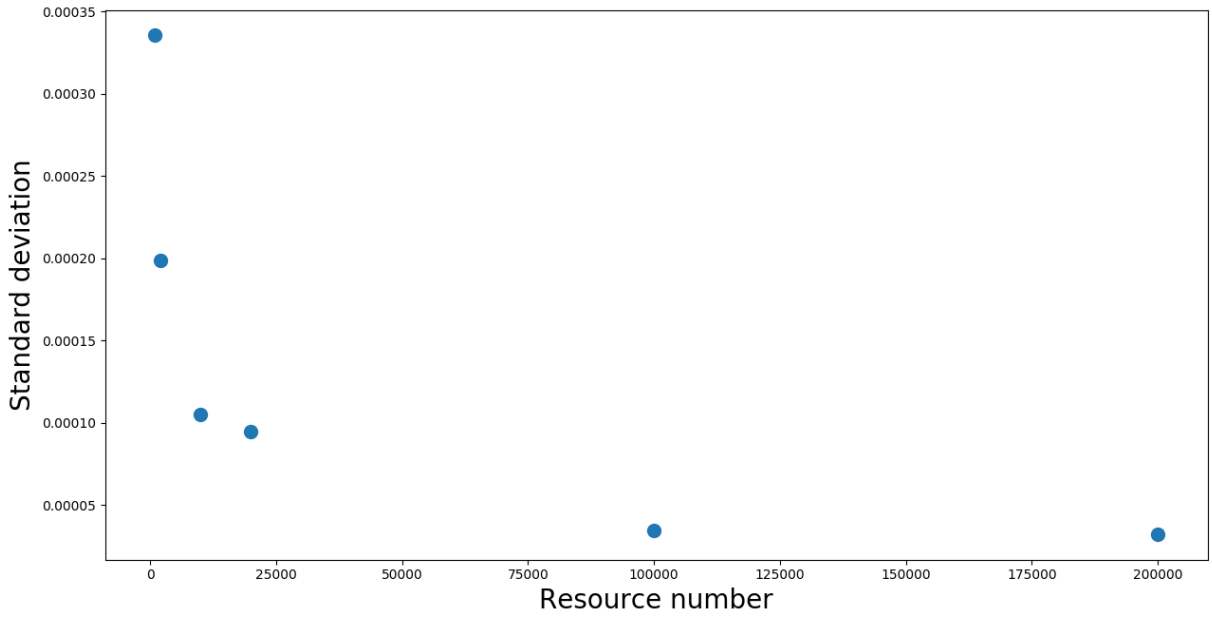

Figure 3.10: The standard deviation of the 10 estimated data points for each resource number for Fig. 3.9.

higher i.e. generate noise maps that are even closer to an identity matrix. This highlights the presence of noise as a serious problem in quantum computation and fortunately the RB protocol is immune to these type of errors.

From Fig. 3.4, it can be seen that for the Clifford gates $n = 3$, the increase of sequence samples K_s has an effect of smoothing out the fluctuation of the data point of the estimates of sequence fidelity. This is evident from the reduction of the mean residuals as K_s increases as shown in Fig. 3.5, which agrees exactly with the core theory of RB where the sequence fidelity converges asymptotically in the limit of

Sequence samples K_s	**Error compared to estimated true value (%)**	
	Clifford	**Non-Clifford**
5	0.01892	0.00245
10	0.01156	0.00477
50	0.00115	0.00265
100	0.00112	0.00215
500	0.00100	0.00015
1000	0.00076	0.00024

Table 3.2: The error percentage, calculated as $\frac{|\bar{F}_{\text{Est}}-\bar{F}_{\text{True}}|}{\bar{F}_{\text{True}}} * 100$, compared with increasing K_s according to Fig. 3.7 and Fig. 3.9.

large number of sequences K_s. The sequence length s in general would not really have a huge impact on the average gate fidelity estimate once it hits the noise floor, hence it was chosen to be 200. This allows unusual fluctuations, if any, to be seen when s becomes higher. The variance of the estimates however do not seem to have changed and are quite consistent. This is likely due to the aforementioned modifications of the canonical RB protocol where the offset is eliminated, and in turn reduces significantly the statistical fluctuation of the variance.

For the investigation of non-Clifford gates, the characteristic exponential decay curve can be observed to have successfully produced in Fig. 3.6, which would first suggest that these specific conjectured non-Clifford t-designs are suitable for performing the RB protocol. From Fig. 3.5, it can be seen that there is no significant decrease of the mean residuals with $K_s = 100$ and above. Hence, in this investigation K_s was chosen to be 200, in order to give a good compromise between the fit accuracy and the run time of the simulation, reducing any unnecessary computational overhead. The increase of gate sizes (higher n) and the fact that they are non-Clifford t-designs does not seem to have any immediate effect compared to when it was performed on the Clifford gates with $n = 3$. In addition, it would seem that these non-Clifford gates are insensitive to the RB protocol since there are hardly any differences between its results compared to when $n = 3$. Moreover, the increase of gate set size does not seem to have any effect on the estimated average gate fidelity $\bar{F}$ since in all six cases the estimates are similar up to 4 s.f. Though it remains inconclusive as to why the gate set size has no impact on any aspect of the simulation, since one might suspect that having a large gate set size might not have allowed randomisation section of the protocol to have twirled over all the gates within that gate set. One explanation might be that the subset of gates from the whole gate set that do end up in the twirl actually forms an approximate t-design and still satisfy the condition in Eq.(2.52). Though it is not immediately clear how the estimates from protocol will be affected in any way as a result. Note that with such a family of gate sets where the number of gates might grow to infinity, there is an off chance one of the gate set might eventually include the conventional non-Clifford T-gate, leaving more to explore about the larger gate sets.

For the fidelity estimates investigation of both Clifford and non-Clifford gates, it can be seen from Fig. 3.8 and Fig. 3.7 that the average gate fidelity converges rapidly to the estimated true value with increasing resource number, with standard deviation following the sharp convergence to a very small value. The error of the estimation is seen in table 3.2 to be as low as 0.00076% for Clifford gates and 0.00015% for non-Clifford gates, with $K_s = 1000$. This agrees with the theory up to an incredible precision.

3.3 Conclusion

In conclusion, the t-design non-Clifford gates of this project have been shown to be compatible with the modified standard Clifford group RB protocol. With the noise crucially assumed to be gate- and time-independent, the numerical analysis has shown by two examples ($n = 3, 50$) that the fidelity of the whole family of gate set can be estimated to a high degree of precision. The non-Clifford gates are shown to be insensitive to the RB protocol. This provides a window of opportunity for future work to investigate how much the RB protocol is constrained by the need of its gate set to form a Clifford group, which can potentially be relaxed. While this did not show whether they possess any unique mathematical properties,

it showed that they can be efficiently characterised by the RB protocol. These non-Clifford gates remain an experimental interest to be implemented with the standard Clifford gates, which may be helpful towards building a universal quantum computer.

Bibliography

[1] D. Ng, "Single Qubit Randomised Benchmarking of Non-Clifford Gates," *Interim Report for MSci Physics Research Project, University of Bristol*, 2020.

[2] R. P. Feynman, "Simulating physics with computers," *International Journal of Theoretical Physics*, vol. 21, pp. 467–488, Jun 1982.

[3] P. W. Shor, "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer," *SIAM Journal on Computing*, vol. 26, p. 1484–1509, Oct 1997.

[4] A. Ekert and R. Jozsa, "Quantum computation and Shor's factoring algorithm," *Rev. Mod. Phys.*, vol. 68, pp. 733–753, Jul 1996.

[5] L. K. Grover, "A fast quantum mechanical algorithm for database search," 1996.

[6] A. Barenco, T. A. Brun, R. Schack, and T. P. Spiller, "Effects of noise on quantum error correction algorithms," *Physical Review A*, vol. 56, p. 1177–1188, Aug 1997.

[7] P. W. Shor, "Scheme for reducing decoherence in quantum computer memory," *Phys. Rev. A*, vol. 52, pp. R2493–R2496, Oct 1995.

[8] A. M. Steane, "Error Correcting Codes in Quantum Theory," *Phys. Rev. Lett.*, vol. 77, pp. 793–797, Jul 1996.

[9] D. P. DiVincenzo and P. W. Shor, "Fault-Tolerant Error Correction with Efficient Quantum Codes," *Phys. Rev. Lett.*, vol. 77, pp. 3260–3263, Oct 1996.

[10] A. Steane, "Multiple-particle interference and quantum error correction," *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, vol. 452, no. 1954, pp. 2551–2577, 1996.

[11] R. Barends, J. Kelly, A. Megrant, A. Veitia, D. Sank, E. Jeffrey, T. White, J. Mutus, A. Fowler, B. Campbell, Y. Chen, Z. Chen, B. Chiaro, A. Dunsworth, C. Neill, P. O'Malley, P. Roushan, A. Vainsencher, J. Wenner, and J. Martinis, "Superconducting quantum circuits at the surface code threshold for fault tolerance," *Nature*, vol. 508, pp. 500–3, 04 2014.

[12] A. Córcoles, E. Magesan, S. Srinivasan, A. Cross, M. Steffen, J. Gambetta, and J. Chow, "Demonstration of a quantum error detection code using a square lattice of four superconducting qubits," *Nature communications*, vol. 6, p. 6979, 04 2015.

[13] N. Ofek, A. Petrenko, R. Heeres, P. Reinhold, Z. Leghtas, B. Vlastakis, Y. Liu, L. Frunzio, S. Girvin, L. Jiang, M. Mirrahimi, M. Devoret, and R. Schoelkopf, "Extending the lifetime of a quantum bit with error correction in superconducting circuits," *Nature*, vol. 536, 07 2016.

[14] I. L. Chuang and M. A. Nielsen, "Prescription for experimental determination of the dynamics of a quantum black box," *Journal of Modern Optics*, vol. 44, p. 2455–2467, Nov 1997.

[15] D. Greenbaum, "Introduction to Quantum Gate Set Tomography," 2015.

[16] A. M. Childs, I. L. Chuang, and D. W. Leung, "Realization of quantum process tomography in NMR," *Phys. Rev. A*, vol. 64, p. 012314, Jun 2001.

[17] J. L. O'Brien, G. J. Pryde, A. Gilchrist, D. F. V. James, N. K. Langford, T. C. Ralph, and A. G. White, "Quantum Process Tomography of a Controlled-NOT Gate," *Phys. Rev. Lett.*, vol. 93, p. 080502, Aug 2004.

[18] M. Riebe, K. Kim, P. Schindler, T. Monz, P. O. Schmidt, T. K. Körber, W. Hänsel, H. Häffner, C. F. Roos, and R. Blatt, "Process Tomography of Ion Trap Quantum Gates," *Phys. Rev. Lett.*, vol. 97, p. 220407, Dec 2006.

[19] J. M. Chow, J. M. Gambetta, L. Tornberg, J. Koch, L. S. Bishop, A. A. Houck, B. R. Johnson, L. Frunzio, S. M. Girvin, and R. J. Schoelkopf, "Randomized Benchmarking and Process Tomography for Gate Errors in a Solid-State Qubit," *Phys. Rev. Lett.*, vol. 102, p. 090502, Mar 2009.

[20] R. C. Bialczak, M. Ansmann, M. Hofheinz, E. Lucero, M. Neeley, A. D. O'Connell, D. Sank, H. Wang, J. Wenner, M. Steffen, and et al., "Quantum process tomography of a universal entangling gate implemented with Josephson phase qubits," *Nature Physics*, vol. 6, p. 409–413, Apr 2010.

[21] C. Stark, "Self-consistent tomography of the state-measurement Gram matrix," *Physical Review A*, vol. 89, May 2014.

[22] C. Stark, "Simultaneous Estimation of Dimension, States and Measurements: Computation of representative density matrices and POVMs," 2012.

[23] S. T. Merkel, J. M. Gambetta, J. A. Smolin, S. Poletto, A. D. Córcoles, B. R. Johnson, C. A. Ryan, and M. Steffen, "Self-consistent quantum process tomography," *Phys. Rev. A*, vol. 87, p. 062119, Jun 2013.

[24] J. F. Poyatos, J. I. Cirac, and P. Zoller, "Complete Characterization of a Quantum Process: The Two-Bit Quantum Gate," *Physical Review Letters*, vol. 78, p. 390–393, Jan 1997.

[25] J. Emerson, R. Alicki, and K. Życzkowski, "Scalable noise estimation with random unitary operators," *Journal of Optics B: Quantum and Semiclassical Optics*, vol. 7, p. S347–S352, Sep 2005.

[26] E. Knill, D. Leibfried, R. Reichle, J. Britton, R. B. Blakestad, J. D. Jost, C. Langer, R. Ozeri, S. Seidelin, and D. J. Wineland, "Randomized benchmarking of quantum gates," *Phys. Rev. A*, vol. 77, p. 012307, Jan 2008. .

[27] E. Magesan, J. M. Gambetta, and J. Emerson, "Scalable and Robust Randomized Benchmarking of Quantum Processes," *Physical Review Letters*, vol. 106, May 2011.

[28] E. Magesan, J. M. Gambetta, and J. Emerson, "Characterizing quantum gates via randomized benchmarking," *Physical Review A*, vol. 85, Apr 2012.

[29] J. P. Gaebler, A. M. Meier, T. R. Tan, R. Bowler, Y. Lin, D. Hanneke, J. D. Jost, J. P. Home, E. Knill, D. Leibfried, and et al., "Randomized Benchmarking of Multiqubit Gates," *Physical Review Letters*, vol. 108, Jun 2012.

[30] E. Magesan, J. M. Gambetta, B. R. Johnson, C. A. Ryan, J. M. Chow, S. T. Merkel, M. P. da Silva, G. A. Keefe, M. B. Rothwell, T. A. Ohki, and et al., "Efficient Measurement of Quantum Gate Error by Interleaved Randomized Benchmarking," *Physical Review Letters*, vol. 109, Aug 2012.

[31] J. J. Wallman, "Randomized benchmarking with gate-dependent noise," *Quantum*, vol. 2, p. 47, Jan 2018.

[32] D. Gottesman, "Stabilizer Codes and Quantum Error Correction," 1997.

[33] S. Bravyi and A. Kitaev, "Universal quantum computation with ideal Clifford gates and noisy ancillas," *Phys. Rev. A*, vol. 71, p. 022316, Feb 2005.

[34] A. M. Meier, B. Eastin, and E. Knill, "Magic-state distillation with the four-qubit code," 2012.

[35] S. Kimmel, M. P. da Silva, C. A. Ryan, B. R. Johnson, and T. Ohki, "Robust Extraction of Tomographic Information via Randomized Benchmarking," *Phys. Rev. X*, vol. 4, p. 011050, Mar 2014.

[36] A. W. Cross, E. Magesan, L. S. Bishop, J. A. Smolin, and J. M. Gambetta, "Scalable randomised benchmarking of non-Clifford gates," *npj Quantum Information*, vol. 2, Apr 2016.

[37] A. Carignan-Dugas, J. J. Wallman, and J. Emerson, "Characterizing universal gate sets via dihedral benchmarking," *Phys. Rev. A*, vol. 92, p. 060302, Dec 2015.

[38] S. Dilkes, "On the structure of a measurement-based unitary 3-design," *Masters dissertation, University of Bristol, Department of Physics*, 2017.

[39] T. Xia, M. Lichtman, K. Maller, A. W. Carr, M. J. Piotrowicz, L. Isenhower, and M. Saffman, "Randomized Benchmarking of Single-Qubit Gates in a 2D Array of Neutral-Atom Qubits," *Phys. Rev. Lett.*, vol. 114, p. 100503, Mar 2015.

[40] J. T. Muhonen, A. Laucht, S. Simmons, J. P. Dehollain, R. Kalra, F. E. Hudson, S. Freer, K. M. Itoh, D. N. Jamieson, J. C. McCallum, A. S. Dzurak, and A. Morello, "Quantifying the quantum gate fidelity of single-atom spin qubits in silicon by randomized benchmarking," *Journal of Physics: Condensed Matter*, vol. 27, p. 154205, mar 2015.

[41] J. Kelly, R. Barends, B. Campbell, Y. Chen, Z. Chen, B. Chiaro, A. Dunsworth, A. Fowler, I.-C. Hoi, E. Jeffrey, and et al., "Optimal Quantum Control Using Randomized Benchmarking," *Physical Review Letters*, vol. 112, Jun 2014.

[42] F. Arute, K. Arya, R. Babbush, D. Bacon, J. Bardin, R. Barends, R. Biswas, S. Boixo, F. Brandao, D. Buell, B. Burkett, Y. Chen, Z. Chen, B. Chiaro, R. Collins, W. Courtney, A. Dunsworth, E. Farhi, B. Foxen, and J. Martinis, "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, pp. 505–510, 10 2019.

[43] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition*. New York, NY, USA: Cambridge University Press, 10th ed., 2011.

[44] A. Rae, *Quantum mechanics*. Taylor & Francis Group, 2008.

[45] *The Physics of Quantum Information: Quantum Cryptography, Quantum Teleportation, Quantum Computation*. Berlin, Heidelberg: Springer-Verlag, 2001.

[46] A. K. Ekert, "Quantum cryptography based on Bell's theorem," *Phys. Rev. Lett.*, vol. 67, pp. 661–663, Aug 1991.

[47] M. D. Eisaman, J. Fan, A. Migdall, and S. V. Polyakov, "Invited Review Article: Single-photon sources and detectors," *Review of Scientific Instruments*, vol. 82, no. 7, p. 071101, 2011.

[48] M. G. A. Paris, "The modern tools of quantum mechanics," *The European Physical Journal Special Topics*, vol. 203, p. 61–86, Apr 2012.

[49] F. Grazioso, "Quantum Measurement Formalism," 2015.

[50] B. Schumacher and M. Westmoreland, *Quantum Processes Systems, and Information*. Cambridge University Press, 2010.

[51] B. Schumacher, "Sending quantum entanglement through noisy channels," 1996.

[52] K. Kraus, A. Böhm, J. Dollard, and W. Wootters, *States, effects, and operations: fundamental notions of quantum theory*. Lecture notes in physics, Springer-Verlag, 1983.

[53] M.-D. Choi, "Completely positive linear maps on complex matrices," *Linear Algebra and its Applications*, vol. 10, no. 3, pp. 285 – 290, 1975.

[54] P. Pechukas, "Reduced Dynamics Need Not Be Completely Positive," *Phys. Rev. Lett.*, vol. 73, pp. 1060–1062, Aug 1994.

[55] Y. S. Weinstein, T. F. Havel, J. Emerson, N. Boulant, M. Saraceno, S. Lloyd, and D. G. Cory, "Quantum process tomography of the quantum Fourier transform," *The Journal of Chemical Physics*, vol. 121, p. 6117–6133, Oct 2004.

[56] M. Barnhill, "Extensions of Randomized Benchmarking," Master's thesis, University of Waterloo, 2015.

[57] M. B. Ruskai, S. Szarek, and E. Werner, "An Analysis of Completely-Positive Trace-Preserving Maps on 2x2 Matrices," 2000.

[58] B. Dirkse, "Statistical Analysis of the Single-qubit Unitarity Randomized Benchmarking Protocol," 05 2017.

[59] C. King and M. B. Ruskai, "Minimal Entropy of States Emerging from Noisy Quantum Channels," 1999.

[60] R. Harper, "Quantum Nescimus: Improving the characterization of quantum systems from limited information," *PhD thesis, University of Sydney*, 2018.

[61] P. S. Turner and D. Markham, "Derandomizing Quantum Circuits with Measurement-Based Unitary Designs," *Physical Review Letters*, vol. 116, May 2016.

[62] R. Solovay and V. Strassen, "A Fast Monte-Carlo Test for Primality," *SIAM Journal on Computing*, vol. 6, no. 1, pp. 84–85, 1977.

[63] Y. Ilan, "Generating randomness: Making the most out of disordering a false order into a real one," *Journal of Translational Medicine*, vol. 17, 12 2019.

[64] M. Adam, "Applications of Unitary k-designs in Quantum Information Processing ," *Masters thesis, Masaryk University Faculty of Informatics*, 2013.

[65] F. G. Brandão, A. W. Harrow, and M. Horodecki, "Efficient Quantum Pseudorandomness," *Physical Review Letters*, vol. 116, Apr 2016.

[66] A. Belovs and J. I. Rosado, "Welch Bounds and Quantum State Tomography," 2008.

[67] A. J. Scott, "Optimizing quantum process tomography with unitary 2-designs," *Journal of Physics A: Mathematical and Theoretical*, vol. 41, p. 055308, Jan 2008.

[68] J. M. Chow, J. M. Gambetta, L. Tornberg, J. Koch, L. S. Bishop, A. A. Houck, B. R. Johnson, L. Frunzio, S. M. Girvin, and R. J. Schoelkopf, "Randomized Benchmarking and Process Tomography for Gate Errors in a Solid-State Qubit," *Physical Review Letters*, vol. 102, Mar 2009.

[69] S. Hadfield, "Quantum pseudorandomness: Constructing unitary t-designs for multiple qubit systems," *Master's Thesis, University of Bristol*, 2016.

[70] C. Dankert, R. Cleve, J. Emerson, and E. Livine, "Exact and approximate unitary 2-designs and their application to fidelity estimation," *Physical Review A*, vol. 80, Jul 2009.

[71] R. Cleve, D. Leung, L. Liu, and C. Wang, "Near-linear constructions of exact unitary 2-designs," 2015.

[72] D. Gross, K. Audenaert, and J. Eisert, "Evenly distributed unitaries: On the structure of unitary designs," *Journal of Mathematical Physics*, vol. 48, p. 052104, May 2007.

[73] M. Ozols, "Clifford Group," 2020.

[74] E. Knill, "Fault-Tolerant Postselected Quantum Computation: Threshold Analysis," 2004.

[75] B. W. Reichardt, "Quantum Universality from Magic States Distillation Applied to CSS Codes," *Quantum Information Processing*, vol. 4, p. 251–264, Aug 2005.

[76] M. M. Wilde, *Quantum Information Theory*. Cambridge University Press, 2013.

[77] M. A. Nielsen, "A simple formula for the average gate fidelity of a quantum dynamical operation," *Physics Letters A*, vol. 303, no. 4, pp. 249 – 252, 2002.

[78] Rudnicki, Z. Puchała, and K. Zyczkowski, "Gauge invariant information concerning quantum channels," *Quantum*, vol. 2, p. 60, Apr 2018.

[79] J. J. Wallman and S. T. Flammia, "Randomized benchmarking with confidence," *New Journal of Physics*, vol. 16, p. 103032, oct 2014.

[80] J. Helsen, J. J. Wallman, S. T. Flammia, and S. Wehner, "Multiqubit Randomized Benchmarking Using Few Samples," *Physical Review A*, vol. 100, Sep 2019.

Appendix

Python Code for Simulations

```
# -*- coding: utf-8 -*-
"""
Created on Sat Sep 20 16:11:04 2019

@author: dn16018

Randomised benchmarking non-Clifford gates for one qubit
"""
import numpy as np
from numpy.linalg import norm
import matplotlib.pyplot as plt
# plt.switch_backend("agg")#switch this on for bluecrystal
import time
from scipy.optimize import curve_fit
import math as m
from mpi4py import MPI

comm = MPI.COMM_WORLD
numtasks = comm.Get_size()
taskid = comm.Get_rank()
MASTER = 0
TAG1 = 1
TAG2 = 2
TAG3 = 3
TAG4 = 4
TAG5 = 5

up_spin = np.array([[1, 0], [0, 0]], dtype=complex)
down_spin = np.array([[0, 0], [0, 1]], dtype=complex)

# Clifford group
II = np.eye(2, dtype=complex)
X = np.array([[0, 1], [1, 0]], dtype=complex)
Y = np.array([[0, -1j], [1j, 0]], dtype=complex)
Z = np.array([[1, 0], [0, -1]], dtype=complex)
PPP = (-II + 1j*X + 1j*Y + 1j*Z)/2
PPM = (-II + 1j*X + 1j*Y - 1j*Z)/2
PMM = (-II + 1j*X - 1j*Y - 1j*Z)/2
MMM = (-II - 1j*X - 1j*Y - 1j*Z)/2
MMP = (-II - 1j*X - 1j*Y + 1j*Z)/2
MPP = (-II - 1j*X + 1j*Y + 1j*Z)/2
PMP = (-II + 1j*X - 1j*Y + 1j*Z)/2
MPM = (-II - 1j*X + 1j*Y - 1j*Z)/2
```

```
cliffords = np.array([II, X, Y, Z, PPP, PPM, PMM, MMM, MMP, MPP, PMP, MPM])
paulis = np.array([II, X, Y, Z])
pauli = m.sqrt(0.5)*paulis
B = (1/np.sqrt(6-2*np.sqrt(3)))*np.array([[1-1j, np.sqrt(3)-1],
                                          [np.sqrt(3)-1, -1-1j]], dtype=complex)


def c_matrix_vander(n: int) -> np.ndarray:
    """
    C_n matrix used to generate the non-Clifford gate sets, generated as a
    Vandermonde matrix.
    for 4*n
    C_n := [[exp(+i*2pi/n), 0],
            [0, exp(-i*2pi/n)]]

    Parameters
    ----------
    n : int
        A number.

    Returns
    -------
    np.concatenate(([II], c_matrix)) : np.ndarray
        A block matrix of C_n matrices.
    """

    exp = np.exp(1j*np.pi/n)
    exp_n = np.array([[exp, 0], [0, exp.conj()]], dtype=complex)
    c_matrix = np.vander(exp_n.ravel(), n,
                         increasing=True)[:, 1:].swapaxes(0, 1).reshape(n-1,
                                                                        2, 2)
    return np.concatenate(([II], c_matrix))


def non_clifford_group(n: int) -> np.ndarray:
    """
    4*n Non-Clifford construction (2-designs)
    e.g. n = 3
    12-Cliffords := {II, X, Y, Z} x {II, D, D^2} where D = B*C_3*B^{dagger}

    for 4*n
    non-Cliffords := {B^{dagger} x {II, X, Y, Z} x B x {II, C_{2n},...,
                     C_{2n}^{n-1}}}

    Parameters
    ----------
    n : int
        A number.

    Returns
    -------
    (conjugated_pauli@c_matrix_vander(n)).reshape(4*n, 2, 2) : ndarray
        A block matrix of 2-designs gate set.
    """

    conjugated_pauli = (B.conj().T@paulis@B).reshape(4, 1, 2, 2)
```

```
    return (conjugated_pauli@c_matrix_vander(n)).reshape(4*n, 2, 2)


def frame_potential(designs: np.ndarray, t: int) -> float:
    """
    Computes the frame potential for t-designs, defined as:
    frame_potential -> (1/K^t)*sum_{i=1, j=1}^n |Tr[U_{i}^{dagger}*U_{j}]|^{2t}

    Parameters
    ----------
    designs : np.ndarray
        (2x2 matrices) A list of t-designs gate set
    t : int
        A number.

    Returns
    -------
    np.sum(np.abs(traced_list)**(2*t))/((len(designs))**t) : float
        The result of computing the frame potential.
    """

    # Computes hermitian tranpose of the t-designs
    designs_herm = designs.transpose(0, 2, 1).conj()

    # Computes the trace of sums of matrix multiplication
    traced_list = np.trace(designs_herm.reshape(len(designs), 1, 2, 2)@designs,
                           axis1=2, axis2=3)
    return np.sum(np.abs(traced_list)**(2*t))/((len(designs))**t)


def pl_rep_operation(matrix: np.ndarray) -> np.ndarray:
    """
    Computes the sum of K_i \\otimes K_i^T for (2x2) matrices.

    Parameters
    ----------
    matrix : np.ndarray
        An array of (2x2) matrices.

    Returns
    -------
    op_sum : np.ndarray
        SUM_i K_i \\otimes K_i^T.
    """
    op_sum = np.zeros((4, 4), dtype=complex)
    for i in range(len(matrix)):
        op_sum += np.kron(matrix[i], matrix[i].conj())
    return op_sum


def init_tensor(input_state: np.ndarray, sample_size: int) -> np.ndarray:
    """
    Given a quantum state(density operator), this function generates an amount
    of it according to "sample_size" into a block matrix/tensor/ndarray, for
    vectorization purposes.
```

```
    Parameters
    ----------
    input_state : np.ndarray
        A density matrix array of shape (2,2).
    sample_size : int
        Number of matrices within the block matrix.

    Returns
    -------
    np.broadcast_to(input_state, (sample_size,)+input_state.shape) : np.ndarray
        An ndarray with dim (sample_size, 2, 2) with input_state as entries.
    """

    return np.broadcast_to(input_state, (sample_size,)+input_state.shape)


def operator_groups2(sample_size: int, n) -> np.ndarray:
    """
    Generates the original Clifford gate set, or Clifford gate set by a basis
    change via matrix B or the non-Clifford gate sets generated by C_n matrix.

    Parameters
    ----------
    sample_size : int
        Size of the ndarray/number of operations running in parallel.
    n : None
        Original Clifford gate set.
    n : "pauli"
        Just the Pauli gate set.
    n : int
        Non Clifford gate set.

    Returns
    -------
    gateset : np.ndarray
        A block matrix of unitary error matrices.
    """

    if n == "pauli":
        return paulis[np.random.choice(paulis.shape[0], sample_size)]
    elif n == "og":
        return cliffords[np.random.choice(cliffords.shape[0], sample_size)]
    else:
        gateset = non_clifford_group(n)
        gateset = np.array([pl_rep_channel(pauli, gateset[i])
                            for i in range(len(gateset))])
        return gateset


def pl_channel(basis: np.ndarray, gate: np.ndarray) -> np.ndarray:
    """
    Convert a channel from 2x2 into 4x4 by a trace-orthonormal basis. When the
    basis is input as the normalised Pauli group, the channel converts into
    Pauli-Liouville representation. Using for loops. Quicker for small
    computations.
```

```
    Parameters
    ----------
    basis : np.ndarray
        An array of the (2x2) normalised Pauli matrices including the Identity.
    gate : np.ndarray
        The channel to be converted.

    Returns
    -------
    channel : np.ndarray
        A channel in Pauli-Liouville representation.
    """

    channel = np.zeros((4, 4), dtype=complex)
    for i in range(len(basis)):
        for j in range(len(basis)):
            channel[i][j] = np.trace(basis[i]@gate@basis[j]@gate.T.conj())
    return channel


def pl_rep_channel(basis: np.ndarray, gate: np.ndarray) -> np.ndarray:
    """
    Convert a channel from 2x2 into 4x4 by a trace-orthonormal basis. When the
    basis is input as the normalised Pauli group, the channel converts into
    Pauli-Liouville representation. Using Numpy vectorisation. Quicker for
    large computations.

    Parameters
    ----------
    basis : np.ndarray
        An array of the (2x2) normalised Pauli matrices including the Identity.
    gate : np.ndarray
        The channel to be converted.

    Returns
    -------
    channel : np.ndarray
        A channel in any/Pauli-Liouville representation.
    """

    basis = np.array([basis, ]*len(basis))
    gate_herm = np.array([[gate.T.conj(), ]*len(basis), ]*len(basis))
    gate = np.array([[gate, ]*len(basis), ]*len(basis))
    return np.trace(basis.transpose(1, 0, 2, 3)@gate@basis@gate_herm, axis1=2, axis2=3)


def pl_ket(basis: np.ndarray, operator: np.ndarray) -> np.ndarray:
    """
    Convert a quantum state in density operator formalism into representation
    of a given trace-orthonormal basis. When the basis is the normalised Pauli
    group, it becomes the Pauli-Liouville representation.

    Parameters
    ----------
    basis : np.ndarray
        An array of the (2x2) normalised Pauli matrices including the Identity.
```

```
    gate : np.ndarray
        The channel to be converted.

    Returns
    -------
    channel : np.ndarray
        A channel in any/Pauli-Liouville representation.
    """

    operator = np.array([operator, ]*len(basis))
    ket = np.trace(pauli@operator, axis1=1, axis2=2)
    return ket.reshape(4, 1)


def gen_truncated(minimum: float, maximum: float, ave: float, sigma: float,
                  max_size: int) -> np.ndarray:
    """
    Generate a bunch of values from a Gaussian distribution with a mean and
    standard deviation. Accept all values that lie within the minimum to
    maximum interval and discard the rest. This effectively makes it a
    truncated Gaussian distribution.

    Parameters
    ----------
    minimum : float
        Minimum value of the accepted interval.
    maximum : float
        Maximum value of the accepted interval.
    ave : float
        Mean of the Gaussian distribution.
    sigma : float
        Standard deviation of the Gaussian distribution.
    max_size : int
        The maximum size of the values that were first drawn from the Gaussian
        distribution.

    Returns
    -------
    einval[index] : np.ndarray
        An array of values generated from the Gaussian that lie within the
        interval [minimum, maximum].

    """

    einval = np.random.normal(ave, sigma, max_size)
    index = (einval > minimum) & (einval < maximum)
    return einval[index]


def get_tau(einval1: float, einval2: float, einval3: float) -> np.ndarray:
    """
    Generate the non-unital part of the Pauli-Liouville noise map

    Parameters
    ----------
    einval1 : float
```

```
        Generated eigenvalue 1.
    einval2 : float
        Generated eigenvalue 2.
    einval3 : float
        Generated eigenvalue 3.

    Returns
    -------
    tau : np.ndarray
        An array of values for tau.
    """

    max_tau = 1 - abs(np.array([einval1, einval2, einval3]))
    tau = max_tau*(2*np.random.uniform(0, 1, (3, max_tau.shape[1]))-1)

    return tau


def q(e: np.ndarray) -> np.ndarray:
    """
    A function the checks part of the constraint that is needed to generate
    the non-unital part of a Pauli-Liouville noise map.

    Parameters
    ----------
    e : np.ndarray
        The previously generated array of eigenvalues.

    Returns
    -------
    np.ndarray
    """

    return (1+e[0, :]+e[1, :]+e[2, :])*(1+e[0, :]-e[1, :]-e[2, :])*\
           (1-e[0, :]+e[1, :]-e[2, :])*(1-e[0, :]-e[1, :]+e[2, :])


def z_eta(t: np.ndarray, l: np.ndarray) -> bool:
    """
    A function the checks part of the constraint that is needed to generate
    the non-unital part of a Pauli-Liouville noise map.

    Parameters
    ----------
    t : np.ndarray
        The previously generated array of tau values.
    l : np.ndarray
        The previously generated array of eigenvalues.

    Returns
    -------
    condition : bool
        The condition of satisfying the constraints.
    """

    norm_t = norm(t, axis=0)
```

```
    sum_term = [(l[i, :]**2)*(2*(t[i, :]**2) - norm_t**2) for i in range(3)]
    condition = (norm_t**4 - 2*norm_t**2 - 2*np.sum(sum_term, axis=0) + q(l))

    return condition


def gen_sigma(minimum: float, maximum: float, ave: float, sigma: float,
              max_size: int, samp_size: int) -> np.ndarray:
    """
    The main function that generates the middle matrix of the Pauli-Liouville
    noise map.

    Parameters
    ----------
    minimum : float
        Minimum value of the accepted interval.
    maximum : float
        Maximum value of the accepted interval.
    ave : float
        Mean of the Gaussian distribution.
    sigma : float
        Standard deviation of the Gaussian distribution.
    max_size : int
        The maximum size of the values that were first drawn from the Gaussian
        distribution.
    samp_size : int
        The amount of values that satisfied all the previous constraints that
        were actually needed.

    Raises
    ------
    ValueError
        Stops working if the amount of values that satisfy the constraints are
        not enough for the amount that's needed.

    Returns
    -------
    sigma_gen : np.ndarray
        An array of many (4x4) such matrices.
    """

    lambda1 = gen_truncated(minimum, maximum, ave, sigma, max_size)
    lambda2 = gen_truncated(minimum, maximum, ave, sigma, max_size)
    if len(lambda1) < len(lambda2):
        lambda2 = lambda2[0:len(lambda1)]
    else:
        lambda1 = lambda1[0:len(lambda2)]

    lambda3 = gen_truncated(minimum, maximum, ave, sigma, max_size)

    if len(lambda1) < len(lambda3):
        lambda3 = lambda3[0:len(lambda1)]
    else:
        lambda2 = lambda2[0:len(lambda3)]
        lambda1 = lambda1[0:len(lambda3)]
```

```
    index = (1+lambda3 > abs(lambda1+lambda2)) & \
            (1-lambda3 > abs(lambda2-lambda1))

    lambda1, lambda2, lambda3 = lambda1[index], lambda2[index], lambda3[index]
    tau = get_tau(lambda1, lambda2, lambda3)
    lambdas = np.array([lambda1, lambda2, lambda3])

    cond = norm(tau, axis=0)**2 < (1 - np.sum(np.square(lambdas), axis=0) +
                                   2*lambda1*lambda2*lambda3)
    index = cond & (z_eta(tau, lambdas) > 0)

    tau = tau[:, index]
    lambda1, lambda2, lambda3 = lambda1[index], lambda2[index], lambda3[index]

    if len(lambda1) <= samp_size:
        raise ValueError("Need to generate a bigger max_size.")
    else:
        pass

    tau = tau[:, 0:samp_size]
    tau = np.vstack((np.ones(tau.shape[1]), tau))
    lambda1 = lambda1[0:samp_size]
    lambda2 = lambda2[0:samp_size]
    lambda3 = lambda3[0:samp_size]
    lambdas = np.array([lambda1, lambda2, lambda3])

    sigma_gen = np.zeros((samp_size, 4, 4))
    sigma_gen[:, range(1, 4), range(1, 4)] = lambdas.T
    sigma_gen[:, :, 0] = tau.T

    return sigma_gen


def create_rotation(samp_size: int, r: float) -> np.ndarray:
    """
    Create the rotation maps in Pauli-Liouville representation that is needed
    to realise a single qubit noise map.

    Parameters
    ----------
    samp_size : int
        The number of matrices needed.
    r : float
        The multiplier the for the small angle generation.

    Returns
    -------
    rotation : np.ndarray
        The (4x4) rotation matrices.
    """

    angles = np.random.normal(0, 1, (3, samp_size))*r
    sin_angles = np.sin(angles)
    cos_angles = np.cos(angles)

    left_angles = np.array([[cos_angles[0], -sin_angles[0]],
```

```
                            [sin_angles[0], cos_angles[0]]])
    mid_angles = np.array([[cos_angles[1], -sin_angles[1]],
                           [sin_angles[1], cos_angles[1]]])
    right_angles = np.array([[cos_angles[2], -sin_angles[2]],
                            [sin_angles[2], cos_angles[2]]])

    left = np.zeros((samp_size, 3, 3))
    left[:, 0:2, 0:2] = left_angles.T
    left[range(0, samp_size), 2:3, 2:3] = np.ones((samp_size, 1, 1))

    mid = np.zeros((samp_size, 3, 3))
    mid[:, 1:3, 1:3] = mid_angles.T
    mid[range(0, samp_size), 0:1, 0:1] = np.ones((samp_size, 1, 1))

    right = np.zeros((samp_size, 3, 3))
    right[:, 0:2, 0:2] = right_angles.T
    right[range(0, samp_size), 2:3, 2:3] = np.ones((samp_size, 1, 1))

    rotation = np.zeros((samp_size, 4, 4))
    rotation[:, 1:4, 1:4] = left@mid@right
    rotation[range(0, samp_size), 0:1, 0:1] = np.ones((samp_size, 1, 1))

    return rotation


def gen_channel(r1: float, r2: float, ave: float, sigma: float, max_size: int,
                samp_size: int) -> np.ndarray:
    """
    Generate the single qubit random CPTP noise map in Pauli-Liouville
    representation where E = U Lambda V, where the U and V are rotations
    in Pauli-Liouville representation.

    Parameters
    ----------
    r1 : float
        The multiplier the for the small angle generation for U.
    r2 : float
        The multiplier the for the small angle generation for V.
    ave : float
        Mean of the Gaussian distribution.
    sigma : float
        Standard deviation of the Gaussian distribution.
    max_size : int
        The maximum size of the values that were first drawn from the Gaussian
        distribution.
    samp_size : int
        The number of (4x4) noise maps needed.

    Returns
    -------
    channel : np.ndarray
        An array of the (4x4) noise maps.
    """

    channel = create_rotation(samp_size, r1)@\
              gen_sigma(0.9, 1, ave, sigma, max_size, samp_size)@\
```

```
                create_rotation(samp_size, r2)

    return channel


def fidelity(channel: np.ndarray) -> np.ndarray:
    """
    Returns the average gate fidelity of a channel in Pauli-Liouville
    representation.

    Parameters
    ----------
    channel : np.ndarray
        The channels.

    Returns
    -------
    TYPE
        An array of all the average gate fidelities.
    """

    diag = np.trace(channel, axis1=1, axis2=2) - 1
    d = m.sqrt(channel.shape[1])
    p = diag/(d**2-1)
    return ((d-1)*p + 1)/d


def unitarity(channel: np.ndarray) -> np.ndarray:
    """
    Returns the unitarity of a channel in Pauli-Liouville representation.

    Parameters
    ----------
    channel : np.ndarray
        The channels.

    Returns
    -------
    TYPE
        An array of all the unitarities.
    """

    d = m.sqrt(channel.shape[1])
    unitals = channel[:, 1:4, 1:4]
    trace = np.trace(unitals.transpose(0,2,1)@unitals, axis1=1, axis2=2)

    return (1/(d**2 - 1))*trace


def unitarity_percent(channel):
    fid = fidelity(channel)
    uni = unitarity(channel)
    d = m.sqrt(channel.shape[1])
    r = 1 - fid
    minU = (1-(d*r)/(d-1))**2
    return (uni-minU)/(1-minU)
```

```
def plotter(fidelity: np.ndarray, unitarity: np.ndarray, xlabel: str,
            ylabel: str):
    """
    Plots unitarity against average gate fidelity

    Parameters
    ----------
    fidelity : np.ndarray
        The average gate fidelities.
    unitarity : TYPE
        The unitarities.
    xlabel : str
        Label of x-axis.
    ylabel : str
        Label of y-axis.
    """

    plt.figure()
    plt.scatter(fidelity, unitarity, s=20, marker="x")
    plt.xlabel(xlabel, fontsize=20)
    plt.ylabel(ylabel, fontsize=20)
    plt.show()


def noisespread():
    """
    Plots the spread of the fidelity of the noise maps as well as their
    estimated errors
    """

    noise = gen_channel(0.06, 0.06, 0.998, 0.04, 30000000, 1000000)
    plt.figure(figsize=(8, 6))
    plt.hist(fidelity(noise), bins = np.linspace(min(fidelity(noise)),
    ↪  max(fidelity(noise)), 50), label="$10^7$ single qubit random noise maps
    ↪  $\\mathbf{\\Lambda}$")
    plt.ylabel("Number of channels generated", fontsize=20)
    plt.xlabel("Fidelity of channel to the Identity Channel", fontsize=20)
    plt.savefig("1", dpi=500)
    plt.legend(fontsize=15)
    plt.figure(figsize=(8, 6))

    ↪  plt.boxplot([(fidelity(noise)-np.average(fidelity(noise)))/np.average(fidelity(noise))],
    ↪  sym="x")
    plt.xlabel("$10^7$ single qubit random noise maps $\\mathbf{\\Lambda}$",
    ↪  fontsize=20)
    plt.ylabel("Estimate error as a fraction \n of average $\\bar{F}$", fontsize=20)
    plt.savefig("2", dpi=500)
    plt.show()


def rb_pauli_liouville(input_state: np.ndarray, seq_len: int,
                       samp_size: int, n: int, max_size: int,
                       input_state_2: np.ndarray) -> float:
    """
```

```
    Computes the sequence fidelity where the calculation of survival
    probabilities are done by Numpy vectorisation and broadcasting.

    Parameters
    ----------
    input_state : np.ndarray
        The first (up) input state in Pauli-Liouville representation.
    seq_len : int
        The sequence length of the RB curve.
    samp_size : int
        The amount of sample size to average the survival probabilties over
        for a particular sequence length.
    n : int
        The number that generates the size of the 4n Clifford and non-Clifford
        gate set.
    max_size : int
        The maximum size drawn from distributions such that samp_size amount
        of them fulfills the CPTP constraints, and generate the noise maps.
    input_state_2 : np.ndarray
        The second (down) input state in Pauli-Lioville representation.

    Returns
    -------
    float
        The sequence fidelity.
    float
        The standard deviation of the sequence fidelity that comes from
        averaging the survival probabilities.
    """

    seq = []

    gateset = operator_groups2(samp_size, n)
    gateset = gateset[np.random.choice(gateset.shape[0], (seq_len, samp_size))]

    prep_noise = gen_channel(0.05, 0.05, 0.955, 0.15, max_size, samp_size)
    prep_noise2 = gen_channel(0.05, 0.05, 0.955, 0.15, max_size, samp_size)

    # input state needs to be 1x4 matrix
    channel1 = prep_noise@init_tensor(input_state, samp_size)
    channel2 = prep_noise2@init_tensor(input_state_2, samp_size)

    unit_noise = np.array([gen_channel(0.06, 0.06, 0.998, 0.04, max_size, samp_size)
    ↪  for i in range(seq_len)])
    unit_noise2 = np.array([gen_channel(0.06, 0.06, 0.998, 0.04, max_size, samp_size)
    ↪  for i in range(seq_len)])

    for j in range(1, seq_len+1):
        q_gates = gateset[j-1, :]
        channel1 = unit_noise[j-1, :]@q_gates@channel1
        channel2 = unit_noise2[j-1, :]@q_gates@channel2

        seq.append(q_gates)

    inverse_gate = np.array(seq[::-1])[0]
    for el in np.array(seq[::-1])[1:]:
```

```
            inverse_gate = inverse_gate@el

        seq_adjoint = inverse_gate.transpose(0, 2, 1).conj()
        adj_noise1 = gen_channel(0.06, 0.06, 0.998, 0.04, max_size, samp_size)
        adj_noise2 = gen_channel(0.06, 0.06, 0.998, 0.04, max_size, samp_size)

        output_state_up = adj_noise1@seq_adjoint@channel1
        output_state_down = adj_noise2@seq_adjoint@channel2

        meas_noise = gen_channel(0.05, 0.05, 0.95, 0.15, max_size, samp_size)
        meas_noise2 = gen_channel(0.05, 0.05, 0.95, 0.15, max_size, samp_size)

        measure_up = init_tensor(m.sqrt(1/2)*np.array([[0, 0, 0, 2]]), samp_size)
        measure_down = init_tensor(m.sqrt(1/2)*np.array([[0, 0, 0, 2]]), samp_size)

        expected_val_up = np.real(measure_up@meas_noise@output_state_up)
        expected_val_down = np.real(measure_down@meas_noise2@output_state_down)

        ave_fid = (expected_val_up-expected_val_down)/2
        ave_fid_error = np.std(ave_fid)

        return np.average(ave_fid), ave_fid_error


def decay(s: float, a: float, f: float) -> float:
    """
    A scaled exponential function for curve fitting specifically for RB.
    y = a*f^s

    Parameters
    ----------
    s : float
        Sequence length.
    a : float
        SPAM parameter.
    f : float
        Depolarising parameter.

    Returns
    -------
    float
        A value.
    """

    return a * (2*f - 1)**s


def get_data2(input_state: np.ndarray, seq_len: int, samp_size: int,
              data_ss: int, n: int, max_size: int, input_state_2: np.ndarray,
              plot=False):
    """
    Generate the sequence fidelity for many different value of sequence length.
    Parallellised by decomposing the for loops so that they can be ran under
    different cores. Assumed a scaling of sqrt(n).

    Parameters
```

```
    ----------
    input_state : np.ndarray
        The first (up) input state in Pauli-Liouville representation.
    seq_len : int
        The max sequence length of the RB curve.
    samp_size : int
        Number of sample size per sequence length for a particular sequence
        fidelity.
    data_ss : int
        The jump in plotted sequence length.
    n : int
        The number that generates the size of the 4n Clifford and non-Clifford
        gate set.
    max_size : int
        The maximum size of the values that were first drawn from the Gaussian
        distribution.
    input_state_2 : np.ndarray
        The second (down) input state in Pauli-Liouville representation.
    plot : bool, optional
        Plots the RB curve if True. The default is False.

    Returns
    -------
    length : np.ndarray
        An array of sequence length values.
    fidelity : np.ndarray
        An array of sequence fidelity for different sequence length values.
    seq_len : int
        Max seqence length.
    error : np.ndarray
        An array of the errors of the sequence fidelity array.
    """

    starttime = time.time()
    length = np.zeros(seq_len, dtype=np.float64)
    fidelity = np.zeros(seq_len, dtype=np.float64)
    error = np.zeros(seq_len, dtype=np.float64)

    if seq_len <= numtasks:
        istart = 1
        iend = istart+seq_len
    else:
        istart = int(seq_len*np.sqrt(taskid/numtasks)) + 1
        iend = int(seq_len*np.sqrt((taskid+1)/numtasks)) + 1
        if iend > seq_len:
            iend = seq_len+1
    print("Rank: ", taskid, " start: ", istart, " end: ", iend)
    for s in range(istart, iend, 1):
        avg_fidelity, fid_error = rb_pauli_liouville(input_state, s, samp_size,
                                                     n, max_size, input_state_2)
        length[s-1] = s
        fidelity[s-1] = avg_fidelity
        error[s-1] = fid_error

    if taskid != MASTER:
        if seq_len <= numtasks:
```

```
            pass
        else:
            offset = istart-1
            chunksize = iend - istart
            comm.send(offset, dest=MASTER, tag=TAG1)
            comm.send(chunksize, dest=MASTER, tag=TAG2)
            comm.Send(length[offset:offset+chunksize], dest=MASTER, tag=TAG3)
            comm.Send(fidelity[offset:offset+chunksize], dest=MASTER, tag=TAG4)
            comm.Send(error[offset:offset+chunksize], dest=MASTER, tag=TAG5)
            timeElapsed = time.time() - starttime
            print("Time elapsed for worker {}: {} seconds".format(taskid, timeElapsed))

    if taskid == MASTER:
        if seq_len <= numtasks:
            pass
        else:
            for source in range(1, numtasks):
                offset = comm.recv(source=source, tag=TAG1)
                chunksize = comm.recv(source=source, tag=TAG2)
                comm.Recv([length[offset:], chunksize, MPI.DOUBLE],
                          source=source, tag=TAG3)
                comm.Recv([fidelity[offset:], chunksize, MPI.DOUBLE],
                          source=source, tag=TAG4)
                comm.Recv([error[offset:], chunksize, MPI.DOUBLE],
                          source=source, tag=TAG5)

        timeElapsed = time.time() - starttime
        print("Time elapsed for Master: {} seconds".format(timeElapsed))

    return length, fidelity, seq_len, error


def fit_curve(x_data: np.ndarray, y_data: np.ndarray, seq_length: int,
              data_ss: int, error: np.ndarray, n: int, plot: bool):
    """
    Curve fitting using SciPy's non-linear least square analysis.

    Parameters
    ----------
    x_data : np.ndarray
        The x-data.
    y_data : np.ndarray
        The y-data.
    seq_length : int
        The max sequence length.
    data_ss : int
        The jump in plotted sequence length.
    error : np.ndarray
        DESCRIPTION.
    n : int
        The number that generates the size of the 4n Clifford and non-Clifford
    plot : bool, optional
        Plots the RB curve if True. The default is False.

    Returns
    -------
```

```
    None.

    """
    popt, pcov = curve_fit(decay, x_data, y_data, bounds=(0, [1., 1.]))

    if plot:
        plt.figure(figsize=(15, 7.5))
        t = np.arange(1, seq_length+data_ss, data_ss)
        fit_error = np.sqrt(np.diag(pcov))
        mean_res = np.average(np.abs(decay(t, *popt) - np.array(y_data)))
        best_fit_vals = tuple(np.vstack((popt, fit_error)).ravel('F'))
        fit_label = 'fit: A=%.10f$\\pm$%.10f\n   ' + '  $\\bar{F}$=%.10f$\\pm$%.10f\n'
        plt.plot(x_data, decay(t, *popt), 'r-',
                 label=fit_label % best_fit_vals)

        plt.errorbar(x_data, y_data, yerr=error, fmt="--x", ecolor='b',
                     elinewidth=1, linewidth=1, markersize=6, capsize=3,
                     label='mean residuals: %7.4f' % (mean_res))
        # plt.title("£n = 1000£, £K_s = 200£", fontsize=20)
        # plt.title("n = " + repr(n) , fontsize=20)
        plt.xlabel("Sequence length $s$", fontsize=20)
        plt.ylabel("Sequence fidelity $F_\\mathbf{G}$", fontsize=20)
        plt.xlim(0, seq_length+1)
        plt.ylim(0, 1)
        plt.legend(shadow=True, fontsize=20)
        plt.savefig("n = " + repr(n) + ".png".format(n))
        plt.show()

    return popt[1]


def variance_plot(exps: int, input_state: np.ndarray, seq_len: int,
                  samp_sizes: np.ndarray, data_ss: int, n: int, max_size: int,
                  input_state_2: np.ndarray, plot: bool):
    """
    Compare the estimates of many average gate fidelity from RB to the direct
    average gate fidelity calculations of many randomly generated noise maps,
    for increasing sequence samples.

    Parameters
    ----------
    exps : int
        The number of RB curves/estimates for a particular sequence sample.
    input_state : np.ndarray
        The first (up) input state in Pauli-Liouville representation.
    seq_len : int
        The max sequence length of the RB curve.
    samp_sizes : np.ndarray
        An array of different sample size for a particular sequence length.
    data_ss : int
        The jump in plotted sequence length.
    n : int
        The number that generates the size of the 4n Clifford and non-Clifford
        gate set.
    max_size : int
        The maximum size of the values that were first drawn from the Gaussian
```

```
        distribution in order to produce the sequence samples required.
    input_state_2 : np.ndarray
        The second (doum) input state in Pauli-Liouville representation.
    plot : bool, optional
        Plots the RB curve if True. The default is False.
    """

    ave_fid_fit_plot = []
    variance = []
    for j in range(len(samp_sizes)):
        ave_fid_fit = []
        for i in range(1, exps+1):
            length, fidelity, seq_length, error = get_data2(input_state,
                                                            seq_len,
                                                            samp_sizes[j],
                                                            data_ss,
                                                            n,
                                                            max_size,
                                                            input_state_2)
            if taskid == MASTER:
                ideal_fid_fit = fit_curve(length, fidelity, seq_length,
                                          data_ss, error, n, plot)
                ave_fid_fit.append(ideal_fid_fit)

        if taskid == MASTER:
            ave_fid_fit_plot.append(np.average(ave_fid_fit))
            variance.append(np.var(ave_fid_fit))

    if taskid == MASTER:
        resources = seq_len * samp_sizes
        plt.figure(figsize=(15, 7.5))
        plt.errorbar(resources, ave_fid_fit_plot, yerr=np.sqrt(variance),
                     fmt="x", ecolor='g', elinewidth=2, capsize=5,
                     markersize=10,
                     label = "RB Estimated $\\bar{F}(\\mathbf{\\mathcal{D}}_{p})$")
        noise = gen_channel(0.06, 0.06, 0.998, 0.04, 30000000, 1000000)
        diag = np.trace(noise, axis1=1, axis2=2) - 1
        d = m.sqrt(noise.shape[1])
        p = diag/(d**2-1)
        fid = np.average(((d-1)*p + 1)/d)
        avg_fid = fid*np.ones(len(resources))
        plt.plot(resources, avg_fid, 'b-', label="$\\bar{F}_{10^7}(\\mathbf{\\Lambda})$
        ↪  = %.10f" % fid)
        print("Average Gate Fidelity: {}".format(ave_fid_fit_plot))

        plt.xlabel("Resource Number ($s \\times K_s$)", fontsize=20)
        plt.ylabel("Average Gate Fidelity $\\bar{F}$", fontsize=20)
        plt.legend(shadow=True, fontsize=20, loc='best')

        plt.savefig('variance.png')  # on for bluecrystal
        plt.show()  # off for bluecrystal

        plt.figure(figsize=(15, 7.5))
        plt.plot(resources, np.sqrt(variance), 'o', markersize=10)
        plt.xlabel("Resource number", fontsize=20)
        plt.ylabel("Standard deviation", fontsize=20)
```

```
        plt.savefig("std.png")  # on for bluecrystal
        plt.show()  # off for bl1uecrystal


starttime = time.time()
input_state = m.sqrt(1/2)*np.array([[1],[0],[0],[1]])
seq_len = 100
samp_size = 100
n = 3
max_size = 100000 # simulation speed scales with max_size, not samp_size
input_state_2 = m.sqrt(1/2)*np.array([[1],[0],[0],[-1]])
data_ss = 1
exps = 2

rb_pauli_liouville(input_state, seq_len, samp_size, n, max_size, input_state_2)
length, fid, seq_len, fid_error = get_data2(input_state, seq_len,
                                            samp_size, data_ss, n,
                                            max_size, input_state_2)

if taskid==MASTER:
    ideal_fid_fit = fit_curve(length, fid, seq_len, 1, fid_error, n, plot=True)

samp_sizes = np.array([100, 200, 300, 400])

variance_plot(exps, input_state, seq_len, samp_sizes, data_ss, n, max_size,
              input_state_2, plot=False)

timeElapsed = time.time() - starttime
print("\nTime elapsed: {} seconds".format(timeElapsed))
```

CERTIFICATION OF OWNERSHIP

Project report presented as part of, and in accordance with, the requirements for the final degree of MSci at the University of Bristol, Faculty of Science.

Author:	Darren Ng
Title:	Single Qubit Randomised Benchmarking of Non-Clifford Gates
Date of Submission:	16/05/2020

Signed: 
Full Name: Darren D Q Ng
Date: 16/05/2020