Practical Advances in Quantum Error Correction & Communication

by

Daniel Benjamin Criger

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Physics

Waterloo, Ontario, Canada, 2013

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Quantum computing exists at the intersection of mathematics, physics, chemistry, and engineering; the main goal of quantum computing is the creation of devices and algorithms which use the properties of quantum mechanics to store, manipulate and measure information. There exist many families of algorithms, which, using non-classical logical operations, can outperform traditional, classical algorithms in terms of memory and processing requirements. In addition, quantum computing devices are fundamentally smaller than classical processors and memory elements; since the physical models governing their performance are applicable on all scales, as opposed to classical logic elements, whose underlying principles rely on the macroscopic nature of the device in question.

Quantum algorithms, for the most part, are predicated on a theory of resources. It is often assumed that quantum computers can be placed in a precise fiducial state prior to computation, and that logical operations are perfect, inducing no error on the system which they affect. These assumptions greatly simplify algorithmic design, but are fundamentally unrealistic. In order to justify their use, it is necessary to develop a framework for using a large number of imperfect devices to simulate the action of a perfect device, with some acceptable probability of failure. This is the study of *fault-tolerant quantum computing*. In order to pursue this study effectively, it is necessary to understand the fundamental nature of generic quantum states and operations, as well as the means by which one can correct quantum errors. Additionally, it is important to attempt to minimize the use of computational resources in achieving error reduction and fault-tolerant computing.

This thesis is concerned with three projects related to the use of error-prone quantum systems to transmit and manipulate information. The first of these is concerned with the use of imperfectly-prepared states in error-correction routines. Using optimal quantum error correction, we are able to deduce a method of partially protecting encoded quantum information against preparation errors prior to encoding, using no additional qubits. The second of these projects details the search for entangled states which can be used to transmit *classical* information over quantum channels at a rate superior to classical states. The third of these projects concerns the transcoding of data from one quantum code into another using few ancillary resources. The descriptions of these projects are preceded by a brief introduction to representations of quantum states and channels, for completeness.

Three techniques of general interest are presented in appendices. The first is an introduction to, and a minor advance in the development of optimal error correction codes. The second is a more efficient means of calculating the action of a quantum channel on a given state, given that the channel acts non-trivially only on a subsystem, rather than the

entire system. Finally, we include documentation on a software package developed to aid the search for quantum transcoding operations.

Acknowledgements

I would like to acknowledge my supervisor, Raymond Laflamme, whose wisdom and patience have been an indispensable aid to the pursuit of knowledge. I would also like to thank my parents, Brock and Janis Criger, for their support and encouragement. The research contained in this thesis would not have been possible without collaborators, including Osama Moussa, Sam Bader, Mary Beth Ruskai, and Chris Granade, whose expertise in computer science and programming has been an inspiration.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Quantum computing is a burgeoning field of research, uniting mathematics, physics, chemistry and engineering, with the eventual aim of producing devices which store and manipulate information using quantum superposition states and unitary operations, in addition to classical states and Boolean logic gates. There are many potential benefits of research into quantum computing, but there are two broad categories in which quantum computing surpasses strictly classical computing:

- Quantum algorithms can be found which outperform classical algorithms. Such algorithms already exist for simulating quantum systems, searching unstructured databases, and performing Fourier transforms, among many other tasks.
- Quantum computers are fundamentally smaller than classical computers, because the essential mechanism of a quantum computer is defined in terms of fundamental microscopic systems, and can be implemented using the smallest known particles.

The simulation of quantum systems alone stands to revolutionize the study of molecular biology, condensed-matter physics, materials science and nanotechnology, among many others; the potential applications of quantum computers are too numerous to list here.

What, then, defines a quantum computer? There is a canonical list of necessary criteria, first laid out by David DiVincenzo in 1997 [26]:

1. There must be well-defined individual quantum systems which store information at the small scale. Typically, these are assumed to be two-level systems, called *qubits*.

2. It must be possible to initialize these qubits into a well-known state before attempting any computation. Typically, this requirement manifests itself as the necessity to cool the system to the ground state of its steady-state Hamiltonian.

3. The qubits must be isolated from their local environment, such that random fluctuations from the external world do not alter the state on the quantum register unpredictably.

4. It must be possible, using an external control system, to drive the system according to an arbitrary Hamiltonian.

5. It must also be possible to measure the final state of the register, in order to interpret the result of a computation.

On the surface, this seems like a reasonable set of requirements. There are two features of these criteria, however, which make them extremely difficult, if not impossible to fulfil exactly.

Firstly, if an external apparatus is used to control the register (requirement 4), the register must interact with it, violating requirement 3, in principle. This conflict in the requirements is more than an abstract, theoretical challenge to quantum computing as defined above; random fluctuations introduced by the control apparatus are a major limiting factor for the overall efficacy of current quantum computing implementations.

The other difficulty in creating a quantum computer can be realized, when examining the thermodynamic implications of requirements 2 and 4. In order to do this, we first introduce the von Neumann entropy, a measure of disorder in quantum states.

1.0.1 The von Neumann Entropy

The quantum state of a system can be represented as a convex sum of orthonormal rank-one projectors:

$$\rho = \sum_j p_j \left|\psi_j\right\rangle\left\langle\psi_j\right|, \; 0 \leq p_j \leq 1 \forall p_j, \; \sum_j p_j = 1. \tag{1.1}$$

The von Neumann entropy is equivalent to the Shannon entropy [74] over the set $\{ p_j \}$:

$$S(\rho) = -\sum_j p_j \log_2 p_j = -\mathrm{tr}(\rho \log_2 \rho) \tag{1.2}$$

Treating $\{ p_j \}$ as a discrete probability distribution, this is the average uncertainty in obtaining a given $|\psi_j\rangle\langle\psi_j|$ upon measurement in the basis defined by $|\psi_j\rangle\langle\psi_j|$.

The thermal equilibrium state for a quantum system with a Hamiltonian $\hat{H}$ is

$$\rho_T = \frac{\exp\left(-\beta\hat{H}\right)}{\mathrm{tr}\left(\exp\left(-\beta\hat{H}\right)\right)},\ \beta = \frac{1}{k_B T} \tag{1.3}$$

To illustrate the difficulty in complying with the second DiVincenzo criterion, we calculate the von Neumann entropy of the thermal state of a simple two-level Hamiltonian:

$$\begin{aligned}
&\hat{H} = \hbar\omega \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \\
&\mathrm{tr}\left(\exp\left(\beta\hat{H}\right)\right) = 2\cosh(\beta\hbar\omega) \\
&\rho_T = \begin{bmatrix} {}^{\exp(-\beta\hbar\omega)}/_{2\cosh(\beta\hbar\omega)} & 0 \\ 0 & {}^{\exp(\beta\hbar\omega)}/_{2\cosh(\beta\hbar\omega)} \end{bmatrix} \\
&S(\rho_T) = -\sum_j p_j \log_2(p_j) = -\left({}^{\exp(-\beta\hbar\omega)}/_{2\cosh(\beta\hbar\omega)}\right)\log_2\left({}^{\exp(-\beta\hbar\omega)}/_{2\cosh(\beta\hbar\omega)}\right) \\
&\qquad\qquad - \left({}^{\exp(\beta\hbar\omega)}/_{2\cosh(\beta\hbar\omega)}\right)\log_2\left({}^{\exp(\beta\hbar\omega)}/_{2\cosh(\beta\hbar\omega)}\right)
\end{aligned} \tag{1.4}$$

The second DiVincenzo criterion from the list above, the requirement that the system can be initialized to a pure state, requires $\beta\hbar\omega = \infty$, necessitating either zero temperature, or an infinite energy gap. Therefore, initialization to a pure state is incompatible with thermal equilibrium. This does not render initialization to a pure state impossible, but difficult in practice.

In addition, the fourth DiVincenzo criterion, the ability to perform arbitrary unitary operations, requires a set of control operations which does not increase the von Neumann entropy of a quantum state. This is infeasible, since it implies the 2$^{\mathrm{nd}}$law of thermodynamics can be saturated, which is physically permissible, but very difficult in practice.

One can contrast this with the case of classical computing, where it is possible to define computational states which are defined over domains. For example, the logical zero on a CMOS-transistor-based logic gate is defined as any voltage between 0 and ${}^{V_{\mathrm{supply}}}/_{2}$, with the logical one being any voltage between ${}^{V_{\mathrm{supply}}}/_{2}$ and V_{supply} [10]. Any probabilistic state

with support only on one of these domains will be treated as an unambiguous logical value, without the necessity for perfect purity at the physical level.

In order to realize the potential of quantum computing as specified by the DiVincenzo criteria, then, we must follow one of two paths. The first is the development of an infinitely-precise method of control over quantum states which are initialized to zero effective temperature, with noiseless measurement to determine the result of a computation. The second is the pursuit of an advanced understanding of the effects of imperfect initialization, control, and measurement, so that a DiVincenzo-compliant quantum computer can be simulated with a high, known probability by a faulty quantum computer. This second path appears more feasible at the outset, since it mimics the development of classical computers, and since the demands it places on experimental apparatus are not as severe as in the first.

This 'second path' is the combination of two areas of research, *quantum information theory* (the study of the effects of general quantum operations on probabilistic quantum states) and *fault-tolerant quantum computing* (the development of ways and means for mitigating the effects of decoherence using quantum algorithms). This thesis is concerned with recent developments in these fields; the remaining chapters correspond to the following topics:

Chapter 2 gives mathematical descriptions which can be used to model quantum systems which interact non-trivially with their local environments.

Chapter 3 is an introduction to current practices in quantum error correction and fault tolerance, with a focus on the most-often-used Pauli stabilizer formalism.

Chapter 4 is an introduction to the specific subset of quantum information theory known as *quantum communication*; the study of quantum information transmitted between distinct entities without mutual simultaneous access.

Chapter 5 details the consequences of relaxing the initialization requirement on a small quantum error-correction protocol.

Chapter 6 details a numerical study of the classical information which can be conveyed through a quantum channel by an entangled state.

Chapter 7 concerns fault-tolerant transformations between quantum error-correcting codes.

Appendix A gives an application of past research into optimal error correction, which is used in Chapter 5 to modify quantum error-correcting codes to protect against initialization noise.

Appendix B gives a more efficient classical algorithm for calculating the effect of a quantum channel on a system, given that the channel can be expressed as a tensor product of small channels.

Appendix C comprises documentation on the QuaEC code library, developed in order to analyse the fault-tolerance of circuits performing transcoding operations.

Chapter 2

Quantum Channels and their Representations

Throughout the remainder of this thesis, we will need to analyse general quantum operations in order to quantify the degree to which a quantum channel allows high-fidelity transmission of classical information (in Chapters 4 and 6), and to determine error-correcting codes which may be used to preserve quantum information, both during storage and computation (in Chapters 3, 5 and 7). To this end, we provide a brief introduction to quantum states and operations. (For an in-depth treatment of quantum states and operations as they pertain to quantum computing, see [46], [60] or [13].) We begin, below, by detailing the 'noiseless' states and operations (pure states and unitary operators), which we wish to simulate with high fidelity in order to implement quantum algorithms (see Chapter 1). We then move on to discuss mixed states and CPTP (completely positive trace-preserving) maps, the 'noisy' quantum states and operations whose properties are integral to the study of quantum communication and error correction. After a discussion of composite systems, and the tensor products required to describe them, we conclude by describing the various mathematical representations of quantum channels. This chapter can be used as a reference for the remainder of the document.

2.1 Pure State Vectors/Projectors

In closed quantum systems (ideal systems not interacting with the rest of the universe), the state is represented by an N-dimensional complex-valued vector with unit length [71]:

$$|\psi\rangle \in \mathbb{C}^N,\ \langle\psi|\psi\rangle = 1, \tag{2.1}$$

where a unit-magnitude complex phase leaves the state invariant, $\exp(i\theta)\,|\psi\rangle \equiv |\psi\rangle$. Given the norm above, there is also a conjugate vector $\langle\psi|$ for any $|\psi\rangle$, it is given by the Hermitian transpose of the state vector:

$$\langle\psi| = (|\psi\rangle)^\dagger. \tag{2.2}$$

Since multiplication by a complex number of unit norm (a 'global phase') leaves the state invariant, the relevant information about $|\psi\rangle$ is contained in $|\psi\rangle\langle\psi|$, since $e^{i\alpha}\,|\psi\rangle\,(e^{i\alpha}\,|\psi\rangle)^\dagger = |\psi\rangle\langle\psi|$. These projectors can also be treated as measurement operators. When a projective measurement $|\phi\rangle\langle\phi|$ is performed on a state $|\psi\rangle$, one of two things occurs:

1. The state $|\phi\rangle$ is output from the measurement with probability $\mathrm{tr}(|\phi\rangle\langle\phi|\,|\psi\rangle\langle\psi|) = |\langle\phi|\psi\rangle|^2$, or
2. The state $\frac{|\psi\rangle - \langle\phi|\psi\rangle|\phi\rangle}{\||\psi\rangle - \langle\phi|\psi\rangle|\phi\rangle\|_2}$ is output from the measurement with probability $1 - |\langle\phi|\psi\rangle|^2$.

There is one measurement of this form which will output the state $|\psi\rangle$ with unit probability; $|\psi\rangle\langle\psi|$. This gives us an operational definition of a pure state, one for which there exists a measurement that will always, with certainty, confirm the state in question.

2.2 Density Matrices

There are, in addition to pure states, states for which no measurement returns a given outcome with certainty. Such a 'mixed' state ρ is represented by a convex sum of pure state projectors:

$$\rho = \sum_j p_j\,|\psi_j\rangle\langle\psi_j| \tag{2.3}$$

The probability of finding a mixed state ρ to be equal to a pure state $|\phi\rangle\langle\phi|$ upon measurement is what one would expect from measuring a pure state projector on an ensemble of differing pure states, or from a single system which is assigned randomly to any of a number of possible states, each with a certain probability:

$$\mathrm{tr}(|\phi\rangle\langle\phi|\,\rho) = \sum_j p_j\,|\langle\phi|\psi_j\rangle|^2 \tag{2.4}$$

Note that density matrices are positive-semidefinite matrices (having $\langle\psi|\rho|\psi\rangle \geq 0 \; \forall \, |\psi\rangle$) with unit trace. We can calculate for each mixed state ρ a measure of its purity, $\mathrm{tr}(\rho^2)$. This purity has a minimal value of $1/d$ in a d-dimensional system, corresponding to the completely mixed state, a uniform distribution over an orthonormal basis. For any rank-one projector σ, $\sigma^2 = \sigma$, therefore $\mathrm{tr}(\sigma^2) = 1$, and these projectors have the maximal value of purity.

2.3 Unitary Operations

We wish to define the set of transformations on quantum states that preserve purity. It is easy to confirm that unitary matrices acting on vectors in the Hilbert space are suited to this task, since unitary matrices preserve the inner product between Hilbert space vectors, and the trace can be taken over any orthonormal basis.

$$U(|\psi\rangle) = U\,|\psi\rangle \;\; \therefore U(\rho) = \sum_j p_j U\,|\psi_j\rangle\langle\psi_j|\,U^\dagger = U\rho U^\dagger \tag{2.5}$$

$$\mathrm{tr}((U\rho U^\dagger)^2) = \mathrm{tr}(U\rho^2 U^\dagger) = \mathrm{tr}(\rho^2) \tag{2.6}$$

Quantum algorithms can be expressed in terms of unitary operations and projective measurements, with computational states corresponding to pure state vectors. We move on to describing more general quantum operations.

2.4 Composite Systems and Tensor Products

General quantum operations need not be represented in terms of the Hilbert space on which the states reside. Quantum operations may act only on a subsystem of the system of interest. It is also possible that such an operation may expand or contract the Hilbert space on which the states reside, by adding or discarding quantum systems. In order to describe these operations, and the constraints which are required on mathematical models which describe them, we discuss composite quantum systems, whose states are described

by matrix tensor products:

$$A \otimes B = \begin{bmatrix} A_{0,0} & \cdots & A_{0,d_A-1} \\ \vdots & \ddots & \vdots \\ A_{d_A-1,0} & \cdots & A_{d_A-1,d_A-1} \end{bmatrix} \otimes \begin{bmatrix} B_{0,0} & \cdots & B_{0,d_B-1} \\ \vdots & \ddots & \vdots \\ B_{d_B-1,0} & \cdots & B_{d_B-1,d_B-1} \end{bmatrix}$$
$$= \begin{bmatrix} A_{0,0}B & \cdots & A_{0,d_A-1}B \\ \vdots & \ddots & \vdots \\ A_{d_A-1,0}B & \cdots & A_{d_A-1,d_A-1}B \end{bmatrix} \tag{2.7}$$

where

$$A_{j,k}B = \begin{bmatrix} A_{j,k}B_{0,0} & \cdots & A_{j,k}B_{0,d_B-1} \\ \vdots & \ddots & \vdots \\ A_{j,k}B_{d_B-1,0} & \cdots & A_{j,k}B_{d_B-1,d_B-1} \end{bmatrix}.$$

If, for example, we wish to describe a combined state which consists of ρ_A on the Hilbert space $\mathcal{H}_A$ and ρ_B on the Hilbert space $\mathcal{H}_B$, we can use $\rho_A \otimes \rho_B$, supported on the combined Hilbert space $\mathcal{H}_{AB}$. Every state on a composite Hilbert space can be written as a sum of tensor products:

$$\rho_{AB} \equiv \sum_{jklm} \rho_{AB,jklm} \left|\psi_{A,j}\right\rangle\left\langle\psi_{A,k}\right| \otimes \left|\phi_{B,l}\right\rangle\left\langle\phi_{B,m}\right|. \tag{2.8}$$

Therefore, the action of a quantum channel can be evaluated when it acts only on a subspace of a composite Hilbert space, or changes the total dimension of the space, considering only the subsystem being acted upon.

2.5 Representations of Quantum Channels

In this section, we discuss constraints on quantum channels by noting that they must output valid density matrices when density matrices are input, even when acting on a subsystem. Operations that preserve non-negativity (the property of a density matrix ρ which guarantees $\langle\psi|\rho|\psi\rangle \geq 0 \forall \left|\psi\right\rangle$) under these conditions are called *completely positive*. Also, since density matrices have unit trace, these operations must be *trace-preserving*. Completely-positive trace-preserving (or CPTP) maps can be expressed in various forms; here, we briefly introduce three representations:

- the Stinespring (or system-environment) representation [86], which treats the action of a quantum channel as the result of a unitary operation acting on a larger Hilbert space,
- the Kraus (or operator-sum) representation [54], which treats the action of a quantum channel as a sum of (in general, non-unitary) operations on the original Hilbert space, and
- the set of 'super-matrix' representations [17], which use (in general, non-unitary) matrices on larger Hilbert spaces to represent the channel.

In order to highlight the different properties of each of these representations, we express thermal equilibration (also known as 'generalized amplitude damping') in the given representation as an example. Generalized amplitude damping is physically-motivated, being a model for random emission and absorption of energy in quantum systems. It is especially applicable in spin-based quantum computing [45] and superconducting-circuit-based quantum computing [16]. There is a simple, operational description for thermal equilibration in terms of discrete-time quantum operations. On two-level systems:

$$\Lambda_{\text{GAD}} : \rho \mapsto \begin{bmatrix} (1-\eta)\rho_{00} + \eta p & \sqrt{1-\eta}\rho_{01} \\ \sqrt{1-\eta}\rho_{01}^* & (1-\eta)(1-\rho_{00}) + \eta(1-p) \end{bmatrix} \tag{2.9}$$

$$\rho_T = \begin{bmatrix} p & 0 \\ 0 & (1-p) \end{bmatrix} \tag{2.10}$$

The fixed state of the generalized amplitude damping channel (Λ_{GAD} above) is labelled ρ_T, it can be thought of as a two-level thermal state for a diagonal Hamiltonian.

We have treated Λ as a function on the space of matrices. In the remainder of this section, we use the CPTP constraints on quantum channels to express them as operators whose action on states can be studied and calculated more efficiently.

2.5.1 The Stinespring Representation

The Stinespring representation is an operational definition of a quantum channel, modelling the action of the channel as the result of an interaction with the external environment which can be described by a large unitary operator:

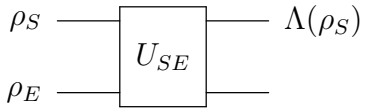


Figure 2.1: The Stinespring representation of a quantum channel Λ. The action of the channel on the state $|\psi\rangle$ is calculated by applying the unitary U_{SE} to the state $\rho_S \otimes \rho_E$, then tracing out the subspace $\mathcal{H}_E$.

It is important to note that, for a given Λ, many sets $\{ \rho_E, U_{SE} \}$ will give identical dynamics on the state ρ_S, and there exists a freedom in which basis one uses to take a 'partial trace' over the subspace $\mathcal{H}_E$. The resulting quantum operation can then be expressed as

$$\Lambda(\rho) = \mathrm{tr}_E(U(\rho_S \otimes \rho_E)U^\dagger). \tag{2.11}$$

Considering the representation-independent definition given in Equation 2.9, we can formulate the quantum circuit implementing this map as a controlled unitary, being controlled by the fixed state ρ_T:

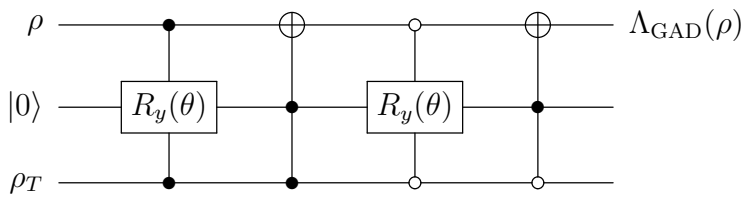


Figure 2.2: A Stinespring representation of the thermal equilibration channel Λ_{GAD}. The action of this channel on the state ρ is, with probability p, to perform a spontaneous emission channel (see [60]), and with probability $1-p$, to perform a spontaneous absorption channel.

There are two more interesting properties that become apparent about quantum channels in the Stinespring representation, and in general. Firstly, any CPTP map acting on density matrices in a d-dimensional Hilbert space can be expressed using at most d^2 elements of the associated Liouville space (the vector space of operators on the Hilbert space). Therefore, it is never necessary to introduce an 'environment' that is larger than two copies of the original system in order to describe the channel Λ acting on ρ. In addition, the initial state of the environment can always be taken to be $|0\rangle\langle 0|$ in the appropriate ancillary Hilbert space, absorbing the parameters contained in the initial state into U_{SE},

using the partial trace after U_{SE} to produce probabilistic mixture. This gives a standard, or canonical, Stinespring representation [60], which we show below for generalized amplitude damping:

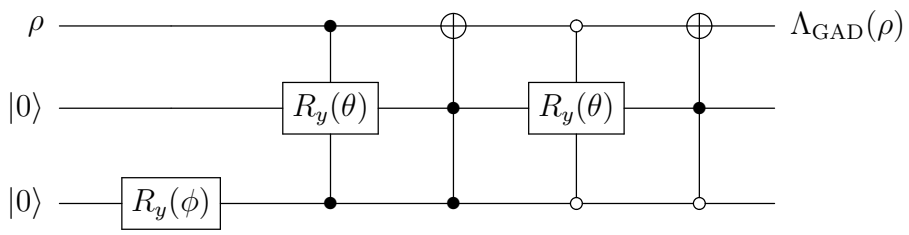


Figure 2.3: A standard Stinespring representation of the circuit producing Λ_{GAD}. Here, y-axis rotations by two angles θ and ϕ are introduced; $\phi = 2\sin^{-1}\left(\sqrt{p}\right)$, $\theta = 2\sin^{-1}\left(\sqrt{\eta}\right)$. Standard constructions such as these are not necessarily concise, but are useful in transforming channels expressed in the Stinespring representation to other representations. (Readers unfamiliar with the circuit diagram notation above may refer to [46].)

The main disadvantage of representing channels in the Stinespring representation is that the resulting operator is d^3-by-d^3 in size. For n-qubit Hilbert spaces, the number of matrix elements grows as 2^{6n}, and the number of elementary operations required to evaluate the action of such a map grows as 2^{9n} in its leading term. This complexity can be reduced with the introduction of more compact, efficient representations of quantum operations. We focus on the Kraus representation and the family of 'super-matrix' representations below.

2.5.2 The Kraus Representation

In order to reduce the size and evaluation time of quantum operations, we simplify the Stinespring quantum operation analytically. In order to do so, we provide a formula for the partial trace introduced in the last subsection:

$$\begin{aligned}\mathrm{tr}_B(\rho_{AB}) &= \sum_j (\hat{\mathbb{1}} \otimes \langle j|)(\rho_{AB})(\hat{\mathbb{1}} \otimes |j\rangle) = \sum_{jklmn} \rho_{AB,klmn} |k\rangle\langle l| \otimes \langle j\,|m\rangle\langle n|\,j\rangle \\ &= \sum_{jkl} \rho_{AB,kljj} |k\rangle\langle l|\,, \end{aligned} \tag{2.12}$$

the partial trace over the system $\mathcal{H}_B$ reduces operators on the composite space $\mathcal{H}_{AB}$ to those on the system $\mathcal{H}_A$, summing over matrix elements whose indices on the system $\mathcal{H}_B$ are diagonal.

$$\begin{aligned}
&\mathrm{tr}_B(\rho_A \otimes \rho_B) = \sum_j (\hat{\mathbb{1}} \otimes \langle j|)(\rho_A \otimes \rho_B)(\hat{\mathbb{1}} \otimes |j\rangle) \\
&= \sum_j \rho_A \otimes \langle j|\rho_B|j\rangle = \mathrm{tr}(\rho_B)\rho_A = \rho_A,
\end{aligned} \tag{2.13}$$

where $|j\rangle$ is a pure state in the Hilbert space $\mathcal{H}_B$.

See above that, for a separable state (expressed as the Kronecker product of two states on smaller Hilbert spaces), the partial trace over one of the subspaces is equivalent to the factor state on the other subspace. Using this formula for the partial trace, we calculate the effect of applying the channel Λ in the standard Stinespring representation.

$$\begin{aligned}
\Lambda(\rho) &= \sum_j \left(\hat{\mathbb{1}} \otimes \langle j|\right) U_{SE} \left(\rho \otimes |0\rangle\langle 0|\right) U_{SE}^\dagger \left(\hat{\mathbb{1}} \otimes |j\rangle\right) \\
&= \sum_j \left(\hat{\mathbb{1}} \otimes \langle j| \cdot U_{SE} \cdot \hat{\mathbb{1}} \otimes |0\rangle\right) \rho \left(\hat{\mathbb{1}} \otimes \langle 0| \cdot U_{SE} \cdot \hat{\mathbb{1}} \otimes |j\rangle\right) \\
&= \sum_j A_j \rho A_j^\dagger;\ A_j = \left(\hat{\mathbb{1}} \otimes \langle j| \cdot U_{SE} \cdot \hat{\mathbb{1}} \otimes |0\rangle\right)
\end{aligned} \tag{2.14}$$

Note that the complete positivity of the unitary U_{SE} and the partial trace defined above guarantees the complete positivity of the channel Λ, the trace of $\Lambda(\rho)$ is

$$\begin{aligned}
&\mathrm{tr}\left(\sum_j A_j \rho A_j^\dagger\right) = \mathrm{tr}\left(\left(\sum_j A_j^\dagger A_j\right)\rho\right) \equiv \mathrm{tr}(\rho) \\
&\therefore \sum_j A_j^\dagger A_j = \hat{\mathbb{1}}.
\end{aligned} \tag{2.15}$$

There is a special class of quantum operations, the unital channels, whose characteristics can easily be specified in the Kraus form. Abstractly, the unitality constraint can be defined as

$$\Lambda\left(\frac{\hat{\mathbb{1}}}{d}\right) = \frac{\hat{\mathbb{1}}}{d}, \tag{2.16}$$

with the constraint being

$$\sum_j A_j A_j^\dagger = \hat{\mathbb{1}} \tag{2.17}$$

in the Kraus representation.

To express generalized amplitude damping in the Kraus form, we calculate $A_j = \left(\hat{\mathbb{1}} \otimes \langle j| \cdot U_{SE} \cdot \hat{\mathbb{1}} \otimes |0\rangle\right)$.

$$U_{SE} = \begin{bmatrix} \tilde{p}\tilde{\eta} & -\sqrt{p}\tilde{\eta} & -\sqrt{\eta}\tilde{p} & \sqrt{p\eta} & 0 & 0 & 0 & 0 \\ \sqrt{p} & \tilde{p} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \tilde{p} & -\sqrt{p} \\ 0 & 0 & 0 & 0 & \sqrt{p\eta} & \sqrt{\eta}\tilde{p} & \sqrt{p}\tilde{\eta} & \tilde{p}\tilde{\eta} \\ 0 & 0 & 0 & 0 & \tilde{p} & -\sqrt{p} & 0 & 0 \\ 0 & 0 & 0 & 0 & \sqrt{p}\tilde{\eta} & \tilde{p}\tilde{\eta} & -\sqrt{p\eta} & -\sqrt{\eta}\tilde{p} \\ \sqrt{\eta}\tilde{p} & -\sqrt{p\eta} & \tilde{p}\tilde{\eta} & -\sqrt{p}\tilde{\eta} & 0 & 0 & 0 & 0 \\ 0 & 0 & \sqrt{p} & \tilde{p} & 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.18}$$

where $\tilde{x} = \sqrt{1-x}$. There are four Kraus operators; $\left(\hat{\mathbb{1}} \otimes \langle 00|\right) U_{SE} \left(\hat{\mathbb{1}} \otimes |00\rangle\right)$, $\left(\hat{\mathbb{1}} \otimes \langle 01|\right) U_{SE} \left(\hat{\mathbb{1}} \otimes |00\rangle\right)$, $\left(\hat{\mathbb{1}} \otimes \langle 10|\right) U_{SE} \left(\hat{\mathbb{1}} \otimes |00\rangle\right)$, and $\left(\hat{\mathbb{1}} \otimes \langle 11|\right) U_{SE} \left(\hat{\mathbb{1}} \otimes |00\rangle\right)$, given below:

$$\begin{aligned} \left(\hat{\mathbb{1}} \otimes \langle 00|\right) U_{SE} \left(\hat{\mathbb{1}} \otimes |00\rangle\right) &= \sqrt{p} \begin{bmatrix} 1 & 0 \\ 0 & \sqrt{1-\eta} \end{bmatrix} \\ \left(\hat{\mathbb{1}} \otimes \langle 01|\right) U_{SE} \left(\hat{\mathbb{1}} \otimes |00\rangle\right) &= \sqrt{1-p} \begin{bmatrix} \sqrt{1-\eta} & 0 \\ 0 & 1 \end{bmatrix} \\ \left(\hat{\mathbb{1}} \otimes \langle 10|\right) U_{SE} \left(\hat{\mathbb{1}} \otimes |00\rangle\right) &= \sqrt{p} \begin{bmatrix} 0 & \sqrt{\eta} \\ 0 & 0 \end{bmatrix} \\ \left(\hat{\mathbb{1}} \otimes \langle 11|\right) U_{SE} \left(\hat{\mathbb{1}} \otimes |00\rangle\right) &= \sqrt{1-p} \begin{bmatrix} 0 & 0 \\ \sqrt{\eta} & 0 \end{bmatrix} \end{aligned} \tag{2.19}$$

The action of Λ on ρ is calculated in the Kraus representation as follows:

$$
\begin{aligned}
\rho &= \begin{bmatrix} \rho_{00} & \rho_{01} \\ \rho_{01}^* & 1-\rho_{00} \end{bmatrix} \\
\Lambda(\rho) &= p \begin{bmatrix} 1 & 0 \\ 0 & \sqrt{1-\eta} \end{bmatrix} \begin{bmatrix} \rho_{00} & \rho_{01} \\ \rho_{01}^* & 1-\rho_{00} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & \sqrt{1-\eta} \end{bmatrix} \\
&+ p \begin{bmatrix} 0 & \sqrt{\eta} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \rho_{00} & \rho_{01} \\ \rho_{01}^* & 1-\rho_{00} \end{bmatrix} \begin{bmatrix} 0 & 0 \\ \sqrt{\eta} & 0 \end{bmatrix} \\
&+ (1-p) \begin{bmatrix} \sqrt{1-\eta} & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \rho_{00} & \rho_{01} \\ \rho_{01}^* & 1-\rho_{00} \end{bmatrix} \begin{bmatrix} \sqrt{1-\eta} & 0 \\ 0 & 1 \end{bmatrix} \\
&+ (1-p) \begin{bmatrix} 0 & 0 \\ \sqrt{\eta} & 0 \end{bmatrix} \begin{bmatrix} \rho_{00} & \rho_{01} \\ \rho_{01}^* & 1-\rho_{00} \end{bmatrix} \begin{bmatrix} 0 & \sqrt{\eta} \\ 0 & 0 \end{bmatrix} \\
&= p \left[\begin{bmatrix} \rho_{00} & \sqrt{1-\eta}\rho_{01} \\ \sqrt{1-\eta}\rho_{01}^* & (1-\eta)(1-\rho_{00}) \end{bmatrix} + \begin{bmatrix} \eta(1-\rho_{00}) & 0 \\ 0 & 0 \end{bmatrix} \right] \\
&+ (1-p) \left[\begin{bmatrix} (1-\eta)\rho_{00} & \sqrt{1-\eta}\rho_{01} \\ \sqrt{1-\eta}\rho_{01}^* & 1-\rho_{00} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & \eta\rho_{00} \end{bmatrix} \right] \\
&= \begin{bmatrix} (1-\eta)\rho_{00} + \eta p & \sqrt{1-\eta}\rho_{01} \\ \sqrt{1-\eta}\rho_{01}^* & (1-\eta)(1-\rho_{00}) + \eta(1-p) \end{bmatrix},
\end{aligned}
\tag{2.20}
$$

which is identical to the action of Λ_{GAD} on ρ shown in 2.9.

Since a quantum operation on a 2^n-dimensional Hilbert space can have 2^{2n} Kraus operators, each of which is a 2^n-by-2^n matrix, applying a quantum operation in Kraus form requires approximately 2^{5n} operations, and 2^{4n} stored parameters. This representation is convenient for applying quantum operations to states analytically; we proceed to describe a family of representations which share this level of convenience, which are also convenient for other applications.

2.5.3 Super-matrix Representations

Since 2^{4n} parameters are required to describe a quantum operation on n qubits, it is interesting to consider the means by which these parameters can be compiled into a single 2^{2n}-by-2^{2n} matrix such that the channel can be easily applied to a state. There are three such representations that are considered here: the Choi matrix, the column-stacked superoperator and the Pauli-basis superoperator.

The Choi Matrix

We examine the effect of a quantum operation on a d-dimensional space, *extended* onto a d^2-dimensional space, acting on a fixed state, defined below:

$$\Phi(\Lambda) = \hat{\mathbb{1}} \otimes \Lambda \left(|\Omega\rangle\langle\Omega| \right); \ |\Omega\rangle = \frac{1}{\sqrt{d}} \sum_{j=0}^{d-1} |j\rangle \otimes |j\rangle \tag{2.21}$$

$$\Phi(\Lambda) = \frac{1}{d}\hat{\mathbb{1}} \otimes \Lambda \left(\sum_{j,k=0}^{d-1} (|j\rangle \otimes |j\rangle)(\langle k| \otimes \langle k|) \right) = \frac{1}{d} \sum_{j,k=0}^{d-1} |j\rangle\langle k| \otimes \Lambda(|j\rangle\langle k|) \tag{2.22}$$

We can write this sum in block matrix notation, noting that each block of the resulting matrix is equivalent to the action of the channel on an elementary matrix:

$$\Phi(\Lambda) = \frac{1}{d} \begin{bmatrix} \Lambda(|0\rangle\langle 0|) & \cdots & \Lambda(|0\rangle\langle d-1|) \\ \vdots & \ddots & \vdots \\ \Lambda(|d-1\rangle\langle 0|) & \cdots & \Lambda(|d-1\rangle\langle d-1|) \end{bmatrix} \tag{2.23}$$

Since the set $|j\rangle\langle k|$ $(0 \leq j, k \leq d-1)$ forms a basis for the space of operators, all the information about Λ is contained within $\Phi(\Lambda)$, and we can express $\Lambda(\rho)$ as

$$\Lambda \left(\sum_{j,k=0}^{d-1} \rho_{jk} |j\rangle\langle k| \right) = \sum_{j,k=0}^{d-1} \rho_{jk} \Lambda(|j\rangle\langle k|) = \sum_{j,k=0}^{d-1} \rho_{jk} (\langle j| \otimes \hat{\mathbb{1}}) \cdot \Phi(\Lambda) \cdot (|k\rangle \otimes \hat{\mathbb{1}}); \tag{2.24}$$

where it is possible to think of $(\langle j| \otimes \hat{\mathbb{1}}) \cdot \Phi(\Lambda) \cdot (|k\rangle \otimes \hat{\mathbb{1}})$ as the block, or submatrix, of the Choi matrix corresponding to the element ρ_{jk} of the matrix ρ.

For generalized amplitude damping, the Choi matrix is:

$$\therefore \Phi(\Lambda) = \begin{bmatrix} 1-\eta+\eta p & 0 & 0 & \sqrt{1-\eta} \\ 0 & \eta(1-p) & 0 & 0 \\ 0 & 0 & \eta p & 0 \\ \sqrt{1-\eta} & 0 & 0 & 1-\eta+\eta(1-p) \end{bmatrix} \tag{2.25}$$

$$\begin{aligned} \Lambda(\rho) &= \rho_{00} \begin{bmatrix} 1-\eta+\eta p & 0 \\ 0 & \eta(1-p) \end{bmatrix} + \rho_{01} \begin{bmatrix} 0 & \sqrt{1-\eta} \\ 0 & 0 \end{bmatrix} \\ &+ \rho_{01}^* \begin{bmatrix} 0 & 0 \\ \sqrt{1-\eta} & 0 \end{bmatrix} + (1-\rho_{00}) \begin{bmatrix} \eta p & 0 \\ 0 & 1-\eta+\eta(1-p) \end{bmatrix} \\ &= \begin{bmatrix} (1-\eta)\rho_{00}+\eta p & \sqrt{1-\eta}\rho_{01} \\ \sqrt{1-\eta}\rho_{01}^* & (1-\eta)(1-\rho_{00})+\eta(1-p) \end{bmatrix} \end{aligned} \tag{2.26}$$

In general, this requires 2^{4n} stored parameters, and 2^{4n} operations to calculate $\Lambda(\rho)$.

Since the matrices $|j\rangle\langle k|$ form a basis for the vector space of density operators, we explore below two other representations for quantum channels which act as matrices on the operator vector space; the column-stacked superoperator, and the Pauli basis superoperator.

The Column-Stacked Superoperator

Since the d-by-d operator space is a vector space, its elements can be expressed as column vectors in a d^2-dimensional space. This can be accomplished for a matrix B, either a density matrix or operator, using the state $|\Omega\rangle$ defined in 2.21:

$$\mathrm{col}(B) = \sqrt{d}(\hat{\mathbb{1}} \otimes B)\,|\Omega\rangle \tag{2.27}$$

In order to obtain a column-stacked superoperator for a given quantum channel, it is convenient to begin with a channel in either the Kraus representation, or the Choi matrix for a channel.

Beginning with the Kraus representation of a channel, the superoperator can be expressed using the well-known identity on generic matrices A, B, and C [40]:

$$\mathrm{col}(ABC) = \left(C^T \otimes A\right)\mathrm{col}(B) \tag{2.28}$$

$$\mathrm{col}\left(\sum_j A_j \rho A_j^\dagger\right) = \sum_j A_j^* \otimes A_j\,\mathrm{col}(\rho),\ \therefore [\Lambda] = \sum_j A_j^* \otimes A_j \tag{2.29}$$

Beginning with the Choi matrix, we can use [40] to obtain the superoperator $[\Lambda]$ for a given channel Λ:

$$[\Lambda] = \sum_{j,k=0}^{d-1} (|j\rangle\langle k| \otimes \hat{\mathbb{1}})\Phi(\Lambda)(\hat{\mathbb{1}} \otimes |j\rangle\langle k|). \tag{2.30}$$

Note that this operation is a permutation of indices, which, if applied twice, results in the identity. This means that the Choi matrix can be obtained using the same formula, given the superoperator.

Either of these methods result in the superoperator for generalized amplitude damping:

$$[\Lambda_{\mathrm{GAD}}] = \begin{bmatrix} 1-\eta+\eta p & 0 & 0 & \eta p \\ 0 & \sqrt{1-\eta} & 0 & 0 \\ 0 & 0 & \sqrt{1-\eta} & 0 \\ \eta(1-p) & 0 & 0 & 1-\eta+\eta(1-p) \end{bmatrix} \tag{2.31}$$

$$[\Lambda_{\mathrm{GAD}}]\,\mathrm{col}(\rho) = \begin{bmatrix} (1-\eta)\rho_{00}+\eta p \\ \sqrt{1-\eta}\rho_{01} \\ \sqrt{1-\eta}\rho_{01}^{*} \\ (1-\eta)(1-\rho_{00})+\eta(1-p) \end{bmatrix} = \mathrm{col}(\Lambda_{\mathrm{GAD}}(\rho)) \tag{2.32}$$

This, too, requires 2^{4n} stored parameters and operations to apply. For certain channels, the superoperator will be sparse in the column-stacked basis. This eases the application of certain channels. Many channels which are dense in this basis are sparse in the Pauli basis, which necessitates the introduction of Pauli-basis superoperators next.

The Pauli-Basis Superoperator

The decomposition of a 2-by-2 density matrix into sums over the basis $|j\rangle\langle k|\,,\ j,k \in \{0,1\}$ is used extensively in the definition of the Choi matrix and column-stacked superoperator. However, it is not the only permissible decomposition; there are other bases which are frequently useful in the study of CPTP maps. The Pauli basis is a good example:

$$\rho = \frac{1}{2}\left(\hat{\mathbb{1}} + \rho_x\sigma_x + \rho_y\sigma_y + \rho_z\sigma_z\right);\ \rho_a = \mathrm{tr}(\rho\sigma_a),\ a \in \{x,y,z\} \tag{2.33}$$

$$\sigma_x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix},\ \sigma_y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix},\ \sigma_z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{2.34}$$

$$\rho \equiv \frac{1}{2}\begin{bmatrix} 1 \\ \rho_x \\ \rho_y \\ \rho_z \end{bmatrix},\ [\Lambda]_{\mathrm{Pauli}} = \left[\begin{array}{c|ccc} 1 & 0 & 0 & 0 \\ \hline \vec{t} & & \mathbf{M} & \end{array}\right] \tag{2.35}$$

where $\vec{t}$ is a 3-by-1 vector of non-unital 'shift' coefficients and $\mathbf{M}$ is a 3-by-3 matrix acting on $[\rho_x,\,\rho_y,\,\rho_z]^T$.

Generalized amplitude damping can be described in terms of its Pauli-basis superoper-

ator as

$$[\Lambda_{\mathrm{GAD}}]_{\mathrm{Pauli}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \sqrt{1-\eta} & 0 & 0 \\ 0 & 0 & \sqrt{1-\eta} & 0 \\ \eta(2p-1) & 0 & 0 & 1-\eta \end{bmatrix} \tag{2.36}$$

$$\begin{aligned} \Lambda_{\mathrm{GAD}}(\rho) &= \frac{1}{2} \begin{bmatrix} 1 \\ \sqrt{1-\eta}\rho_x \\ \sqrt{1-\eta}\rho_y \\ \eta(2p-1) + (1-\eta)\rho_z \end{bmatrix} \\ &= \frac{1}{2}(\hat{\mathbb{1}} + \sqrt{1-\eta}(\rho_x\sigma_x + \rho_y\sigma_y) + (\eta(2p-1) + (1-\eta)\rho_z)\sigma_z) \\ &= \begin{bmatrix} (1-\eta)\rho_{00} + \eta p & \sqrt{1-\eta}\rho_{01} \\ \sqrt{1-\eta}\rho_{01}^* & (1-\eta)(1-\rho_{00}) + \eta(1-p) \end{bmatrix}. \end{aligned} \tag{2.37}$$

Many channels on qubits exhibit the same symmetries as generalized amplitude damping, requiring only four stored parameters, and greatly simplifying the calculation of $\Lambda(\rho)$. Generally, a Pauli-basis superoperator requires the same number of stored parameters and the same amount of calculation in applying them to states, 2^{4n} parameters/operations.

2.6 Conclusion

We have seen parametrizations for pure and mixed states, as well as quantum channels both perfect (unitary) and imperfect (CPTP). Mixed states and CPTP channels are ubiquitous, and the study of quantum communication/error-correction makes extensive use of them. In the following two chapters, we outline quantum error-correction and communication.

Chapter 3

Quantum Error Correction, the Stabilizer Formalism, and Fault Tolerance

In the previous chapter, we discussed non-unitary quantum operations, and gave examples of common non-unitary processes in quantum systems. While these non-unitary operations can be useful during initialization and measurement of quantum systems, they are detrimental to stable memory and high-fidelity control (DiVincenzo criteria 3 and 4, see Chapter 1). In order to simulate unitary evolution, it is necessary to use a set of states entangled across multiple imperfect quantum systems. This is the practice of *quantum error correction* [25]. In this chapter, we describe the basic properties of quantum codes and criteria for correcting errors on quantum states, which is the prior work necessary for the introduction of augmented error correction codes in Chapter 5 and quantum transcoding in Chapter 7.

3.1 Error-Correcting Codes

In any information-processing system, it is possible that some interaction with the environment will result in the application of an unwanted operation, regardless of whether the system is classical or quantum-mechanical. Such operations are called *errors*. A common classical error model is symmetric bit-flip, which takes an input state, labelled α, to the state $1 - \alpha$ with probability p, for $\alpha \in \{ 0, 1 \}$, $0 \leq p \leq 1$:

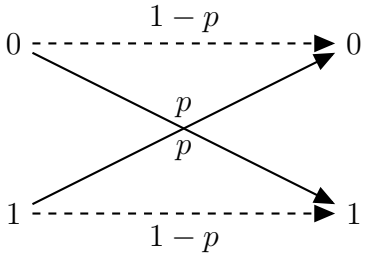


Figure 3.1: Symmetric bit-flip. With probability p, an incoming bit is 'flipped', changing its state from 0 to 1 and vice versa.

Given access to multiple bits which are subject to errors, individual states which remain distinguishable after being subjected to the error in question can be found. We refer to this set of resilient states collectively as the *error-correcting code*, with each state also being known as a *codeword*. For example, the states 000 and 111 on a three-bit register remain distinguishable after a single bit-flip error, since individual bit flips correspond to steps on the following graph:

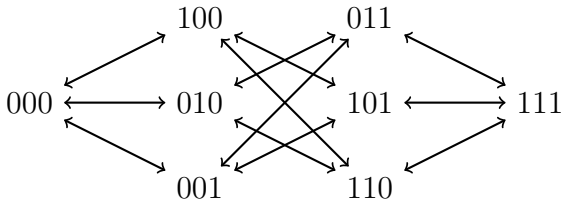


Figure 3.2: States on three bits which can be transformed to one another by single bit flip errors.

Note that the state 000 can be mapped by a single probabilistic bit flip to any state from the set $\{\, 000, 100, 010, 001 \,\}$, where the state 111 is mapped to its complement. Therefore, if only one bit flip occurs, the original state from the set of codewords $\{\, 000, 111 \,\}$ can be recovered, knowing that the other six states correspond to errors. The probability of successful recovery is $(1-p)^3 + 3p(1-p)^2$, which is greater than $1-p$, the probability of preserving the original message without encoding, for $p < 1/2$.

A similar procedure can be used to protect quantum information against bit-flip errors, mapping the state $|0\rangle$ to $|000\rangle$ and $|1\rangle$ to $|111\rangle$. The use of this three-bit code to correct bit-flip is seen in the circuit below:

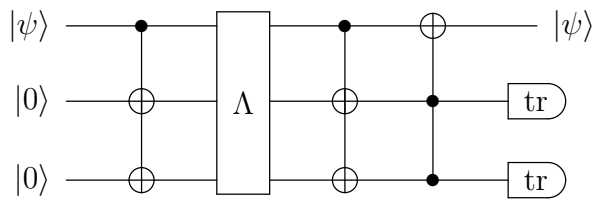


Figure 3.3: An error-correcting circuit for the three-qubit channel Λ, whose Kraus operators are proportional to $\hat{\mathbb{1}} \otimes \hat{\mathbb{1}} \otimes \hat{\mathbb{1}} = \hat{\mathbb{1}}^{\otimes 3}$, $X \otimes \hat{\mathbb{1}} \otimes \hat{\mathbb{1}} = X_1$, $\hat{\mathbb{1}} \otimes X \otimes \hat{\mathbb{1}} = X_2$, and $\hat{\mathbb{1}} \otimes \hat{\mathbb{1}} \otimes X = X_3$. See [46], Chapter 10.

$|\psi\rangle$, above, is an arbitrary one-qubit input state, which can be expressed as a linear combination $\alpha |0\rangle + \beta |1\rangle$. The addition of two qubits in the state $|0\rangle$, followed by the controlled $X \otimes X$ gate results in the arbitrary encoded state $\alpha |000\rangle + \beta |111\rangle$. This state can be written as a linear combination of $|000\rangle$ and $|111\rangle$; the codewords of the repetition code. To show how the circuit corrects these errors, we describe the role of the Toffoli gate at the end of the circuit. The four possible states after the second controlled-$X \otimes X$ gate are:

Error	Resulting State
$\hat{\mathbb{1}}^{\otimes 3}$	$\alpha \|000\rangle + \beta \|100\rangle = \|\psi\rangle \otimes \|00\rangle$
X_1	$\alpha \|111\rangle + \beta \|011\rangle = (X \|\psi\rangle) \otimes \|11\rangle$
X_2	$\alpha \|010\rangle + \beta \|110\rangle = \|\psi\rangle \otimes \|10\rangle$
X_3	$\alpha \|001\rangle + \beta \|101\rangle = \|\psi\rangle \otimes \|01\rangle$

Table 3.1: States resulting from the application of single bit-flip errors to the encoded state $\alpha |000\rangle + \beta |111\rangle$, followed by the decoding operation. Note that only the error X_1 produces an error on the output state, and it is uniquely associated with the state $|11\rangle$ on the ancillae, regardless of the encoded state.

These states are each tensor products of two components, the second of which is commonly called the *error syndrome*. This syndrome identifies the errors from the map, so that the recovery operation (in this case, the final Toffoli gate), can apply the appropriate correction operator (in this case, the Pauli X) to states with specific syndromes (in this case, $|11\rangle$). Note that it is not necessary to apply a Toffoli gate to correct the indicated error, the syndrome bits can be measured, and classical control can be used to correct the resulting error (see [83, 27, 5] for examples of error-correction protocols which use classical control).

Thus, for a specified error model, a code can be found which allows information about

the logical state to be recovered after errors have acted on the system, given that the output states of the error channel are distinguishable. In the next section, we discuss a general requirement for the recovery of information from noisy channels.

3.2 The Knill-Laflamme Criterion

The requirement that codewords transform under the error map to sets of distinguishable states can be expressed mathematically, for a generic set of codewords $\{\ |\psi_j\rangle\ \}$ and error operators $\{\, E_a \,\}$. This is the Knill-Laflamme criterion [51]:

$$\langle\psi_j|E_a^\dagger E_b|\psi_k\rangle = C_{ab}\delta_{jk}. \tag{3.1}$$

When it is satisfied, the existence of a recovery operator (such as the Toffoli gate above) is implied. Note that the value of the inner product when $j = k$ is independent of j, so that there is no correlation between the codeword sent through the channel and the coefficient C_{ab}. For the three-bit code correcting a single bit-flip, the Knill-Laflamme criterion is satisfied, since the states in the set $\{|000\rangle, |100\rangle, |010\rangle, |001\rangle\}$, are orthogonal to the states in the set $\{|111\rangle, |011\rangle, |101\rangle, |110\rangle\}$, corresponding exactly with the classical repetition code correcting bit-flip.

This exact criterion for whether error sets are correctable provides a rigorous, convenient means of qualifying the performance of an error-correcting code. However, the Knill-Laflamme criterion is rarely satisfied exactly for physically-motivated noise maps. We will discuss, in the following section, we discuss a means of quantifying the ability of a quantum channel to preserve or destroy quantum information when the Knill-Laflamme criterion is not satisfied exactly.

3.3 Channel Fidelity

It is rarely, if ever, possible to design a quantum error-correcting code which can perfectly correct a physically-motivated error channel. It is therefore important to evaluate the performance of such a given code using numerical metrics. One such metric is a specific case of Schumacher's entanglement fidelity [72], typically called the *channel fidelity*:

$$\begin{aligned} F_C(\Lambda) &= \langle\Omega|\Phi_\Lambda|\Omega\rangle\,, \\ \Phi_\Lambda &= \hat{\mathbb{1}} \otimes \Lambda(|\Omega\rangle\langle\Omega|) \end{aligned} \tag{3.2}$$

Here, $|\Omega\rangle$ is defined as:

$$|\Omega\rangle = \frac{1}{\sqrt{d}} \sum_{j=0}^{d-1} |j\rangle \otimes |j\rangle \,. \tag{3.3}$$

Note that this fidelity is linear with respect to the Choi matrix Φ_Λ, which allows for efficient optimization over families of channels (for more details, see [68], Chapter 5 of this thesis, or Appendix A), and it has an operational definition, in that it is proportional to the average state fidelity of the output of the channel with the input, the expression $\int \langle\psi|\Lambda(|\psi\rangle\langle\psi|)|\psi\rangle \, d\psi$ [44]. The channel fidelity can also be calculated given the Kraus operators $\{ A_j \}$, although it is not linear in the Kraus representation:

$$F_C(\Lambda) = \sum_j |\mathrm{tr}(A_j)|^2 \,, \quad \sum_j |\mathrm{tr}(A_j + B_j)|^2 \neq \sum_j |\mathrm{tr}(A_j)|^2 + \sum_j |\mathrm{tr}(B_j)|^2 \tag{3.4}$$

There is a family of methods for deriving quantum error correction codes which rely on violating the Knill-Laflamme criterion, and instead maximizing the channel fidelity of the corrected noise channel [67, 30, 29]. Codes derived using these methods can perform more effectively than the exact codes presented above, at the cost of increased complexity [56, 18].

It is interesting to note that the 'exact' codes are also approximate, in the sense that they correct an approximation to some physically-motivated noise. Returning to the bit-flip code presented earlier, we note that it will exactly correct the error set $\{ \hat{\mathbb{1}}^{\otimes 3}, X_1, X_2, X_3 \}$. This error set is unlikely to occur if errors on independent qubits have independent probabilities, since, in this channel, if the operator X has been applied to one of the qubits, we are guaranteed that it has not been applied to the others. A more realistic model is an independent noise map on each qubit:

$$\Lambda_{\mathrm{FLIP}} = \left\{ \sqrt{1-p}\,\hat{\mathbb{1}},\ \sqrt{p}\,X \right\} \tag{3.5}$$

$$\Lambda_{\mathrm{FLIP}}^{\otimes 3} = \Big\{ \sqrt{1-p}^{\,3}\,\hat{\mathbb{1}}\hat{\mathbb{1}}\hat{\mathbb{1}},\ (1-p)\sqrt{p}\,X\hat{\mathbb{1}}\hat{\mathbb{1}},\ (1-p)\sqrt{p}\,\hat{\mathbb{1}}X\hat{\mathbb{1}},\ (1-p)\sqrt{p}\,\hat{\mathbb{1}}\hat{\mathbb{1}}X,$$
$$p\sqrt{1-p}\,\hat{\mathbb{1}}XX,\ p\sqrt{1-p}\,X\hat{\mathbb{1}}X,\ p\sqrt{1-p}\,XX\hat{\mathbb{1}},\ \sqrt{p}^{\,3}\,XXX \Big\} \tag{3.6}$$

Note that, if the bit-flip probability p is small, $(1-p)\sqrt{p}$ is much larger than $p\sqrt{1-p}$, so the channel can be approximated by one with Kraus operators proportional to those which can be corrected exactly by the repetition code above. In this sense, all error correction can be thought of as approximate.

We can think of an error-correcting code as being useful when the channel fidelity of the corrected noise channel exceeds that of the uncorrected noise acting on a single bit. For example, if the bit-flip channel is left uncorrected, the channel fidelity is

$$F_C(\Lambda_{\text{FLIP}}) = 1 - p. \tag{3.7}$$

The channel fidelity for bit-flip, corrected using the three-qubit code, is

$$F_C(\Lambda_3) = 1 - 3p^2 + 2p^3. \tag{3.8}$$

When $0 < p < 1/2$, this code can be considered useful, since $F_C(\Lambda_3) > F_C(\Lambda_{\text{FLIP}})$ in this regime.

3.3.1 The Pauli Basis for Errors

We have seen that, given an error model which is known and specified prior to encoding, it is possible to create a quantum error-correcting code which allows information to be recovered when the system is subject to errors, and to quantify its performance. This ability to correct errors extends to the case in which the error model is not completely specified, allowing for codes which correct classes of errors. This is important, since it increases the versatility of quantum error-correcting codes, allowing errors to be corrected without a perfect error model.

It is possible to correct more general classes of errors because, for any code correcting the error set $\{E_a\}$, any error set $\{F_b\}$ will be correctable using the same code, if $F_b = \sum_a c_{ab} E_a \, \forall F_b$, for some complex coefficients c_{ab} (for a comprehensive proof, see [60], Chapter 10). For this reason, we require a code that corrects a *basis* of the possible errors within a specific class in order to state with certainty that it can correct *any* error within that class. The most popular basis for errors is the basis of Pauli matrices, defined in Equation 2.34.

Given a code which can correct a Pauli error on one qubit (called a *weight-one* Pauli error), one can correct any single-qubit error. If errors are the result of the random application of unwanted operators with some probability p_{err}, then the probability of a two-qubit error is proportional to $p_{\text{err}}^2 < p_{\text{err}}$, and the use of codes to correct weight-one errors is well-justified. In the following sections, we will discuss an efficient means of constructing error-correcting codes, given the constraint that the error set is composed of Pauli operators. We will then show how this is extended to allow computation with imperfect control operations, and point out some of the limitations of this approach.

3.4 The Stabilizer Formalism

Error-correcting codes are difficult to derive by using the Knill-Laflamme condition, and their performance is difficult to evaluate using the channel fidelity, because, for an n-qubit quantum code, the space of possible codewords over which to search or calculate grows as 2^n. Fortunately, the symmetries of the Pauli group can be used to design quantum codes to correct Pauli errors, using only a quadratic number of parameters. This is the stabilizer formalism [36], which we discuss in this section. This formalism is best understood with a few properties of the n-qubit Pauli group, which we detail below.

3.4.1 The n-Qubit Pauli Group

The n-qubit Pauli group is a finite matrix group consisting of tensor products of the following matrices:

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \quad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

Table 3.2: Matrix representations of the one-qubit Pauli operators.

Note that the usual tensor product notation for such an operator (e.g. $X \otimes Y \otimes Z \otimes I \otimes Z$) is often eschewed in favour of a more concise product notation (e.g. $XYZIZ$). There are many special properties of the n-qubit Pauli group which are relevant to quantum error correction in the stabilizer formalism. In this section, we discuss a few of the Pauli group's special properties.

Firstly, the product of two n-qubit Paulis can be determined without explicit matrix multiplication. This is due to the closure of the single-qubit group under multiplication, and a convenient property of the tensor product:

$$(A \otimes B)(C \otimes D) = AC \otimes BD.$$

For example, $(XYZ)(ZIX) = XZ \otimes YI \otimes ZX = (-iY) \otimes Y \otimes (iY) = YYY$.

Also, each pair of n-qubit Pauli operators either commutes or anti-commutes. Commutation of a pair of n-qubit Pauli operators p and q can be easily determined by examining the commutation of their single-qubit constituents. If an even number of these constituents anti-commute, then p and q commute. For example, $[XYZ, ZIX] = 0$ since $\{X, Z\} = 0$, and two of the single-qubit constituents of Paulis XYZ and ZIX anti-commute.

There is a third special property of the elements of the Pauli group on n qubits which allows us to specify an $(n-1)$-qubit subspace of a Hilbert space using a single Pauli operator. Namely, each n-qubit Pauli has 2^{n-1} eigenvalues equal to 1 and 2^{n-1} eigenvalues equal to -1, so labelling the $+1$-eigenspace of a Pauli P as a subspace of interest results in a subspace of dimension 2^{n-1}, specified using a Pauli with n characters in its description. If a second Pauli is chosen, such that it commutes with, and is not equal to, the first, then the two share an eigenbasis, and their *mutual* $+1$-eigenspace is 2^{n-2}-dimensional. In general, selecting m Paulis which mutually commute and are *independent* (no member of the set being a product of the others) results in a subspace which is 2^{n-m}-dimensional. This property of the Pauli group is what motivates the definition of a stabilizer code in the following section.

3.5 Stabilizer Codes

As with any error-correcting code, a stabilizer code is represented by the set of its codewords C:

$$C = \{|\psi\rangle \mid P|\psi\rangle = |\psi\rangle \, \forall P \in S\}. \tag{3.9}$$

Here, S is the stabilizer, a mutually-commuting set of 2^{n-k} Paulis, defining a code which protects k-qubit states. It deserves mention that, if two Paulis P_1 and P_2 are in the stabilizer for a given code, so is P_1P_2 (since $P_1P_2|\psi\rangle = P_1|\psi\rangle = |\psi\rangle$), so it is sufficient to specify a *generating set* of $n-k$ Paulis for the stabilizer.

Returning to the example of protecting a quantum state against bit-flip using a three-qubit repetition code, we can express the stabilizer as

$$S = \langle ZZI, ZIZ \rangle, \tag{3.10}$$

with $\langle A \rangle$ specifying the group generated by the set A. This definition of the code is more concise than the circuit representation given in Figure 3.3, requiring only $n(n-k)$ symbols to represent, as opposed to the 2^{2n} parameters required to specify the encoding operator explicitly. It is advantageous, therefore, to work with the stabilizer whenever possible.

It is possible to determine which errors can be corrected by a stabilizer code using only the Paulis describing the generating set. To see this, we first note that specifying a generator for a code divides the Pauli group on n qubits into three categories:

1. Those that are in the stabilizer S. Note that, for any Paulis P_1 and P_2 in S, $\langle\psi_j|P_1P_2|\psi_k\rangle = \langle\psi_j|\psi_k\rangle$, trivially. Also, any Pauli drawn from the stabilizer will have no effect on the logical state, since $P|\psi\rangle = |\psi\rangle$.

2. Those that anti-commute with at least one element of the stabilizer. Choosing S_* to be one of the stabilizer generators with which the a Pauli P anti-commutes, we can write

$$\begin{aligned}&\langle\psi_j|P|\psi_k\rangle = \langle\psi_j|S_*PS_*|\psi_k\rangle \\ &= -\langle\psi_j|S_*S_*P|\psi_k\rangle = -\langle\psi_j|P|\psi_k\rangle\,, \quad \therefore \langle\psi_j|P|\psi_k\rangle = 0.\end{aligned} \tag{3.11}$$

3. Those that are not within S, but commute with every element of S. This set of Paulis is typically called the *normalizer, mod* S, or $N(S)\backslash S$ of the stabilizer code. If N is such a Pauli, then $|\nu\rangle = N|\psi_j\rangle$ is a codeword, since $P|\nu\rangle = PN|\psi\rangle = NP|\psi\rangle = N|\psi\rangle = |\nu\rangle$ for all P in the stabilizer. Therefore, for some state $|\psi_j\rangle$, $\langle\psi_j|N|\psi_j\rangle \neq 1$. The subset $N(S)\backslash S$ is the set of logical Pauli operators.

The three-qubit bit-flip code is simple enough that we can explicitly calculate these sets of Paulis. Firstly, we generate the stabilizer:

$$S = \langle ZZI,\, ZIZ\rangle = \{\, III, ZZI, ZIZ, IZZ \,\} \tag{3.12}$$

The normalizer, mod S of the bit-flip code can be expressed in terms of *cosets* of the stabilizer, which are sets of stabilizer elements, multiplied by a specific operator:

$$p \cdot S = \{\, ps \,|\, s \in S \,\}\,. \tag{3.13}$$

The normalizer, mod S of the bit-flip code consists of the stabilizer cosets of XXX (which, for clarity, is $\{\, XXX, -YYX, -YXY, -XYY \,\}$), YYY, and ZZZ. The Paulis which anti-commute with at least one element of the stabilizer, can, in turn be expressed as cosets of the *normalizer*, with the coset operators being XII, IXI, and IIX, the errors which the code is designed to correct.

Errors are detected by measuring the stabilizer generators, the measurement results providing a *syndrome* which can reveal the nature of an error without yielding any information about the codeword. For the bit-flip code, these measurement results can be easily tabulated:

Error	$\langle ZZI \rangle$	$\langle ZIZ \rangle$
III	1	1
XII	-1	-1
IXI	-1	1
IIX	1	-1

Table 3.3: Measurement results for the stabilizer generators of the three-qubit bit-flip code, in the cases where either no error has occurred, or a single-qubit bit-flip error has occurred.

These measurement results can then be used to implement a classically-controlled Pauli, which corrects the detected error.

Stabilizer codes, then, can fulfil the Knill-Laflamme criterion for any set of errors $\{E\}$ for which $E_a E_b$ is never in $N(S) \backslash S$. For codes correcting weight-one Pauli errors on n qubits, the set $E_a E_b$ is the set of all n-qubit Paulis of weight one or two (recall, from Subsection 3.3.1 that, to correct general single-qubit errors, it is sufficient to correct single-qubit Pauli errors). The minimum weight of a logical Pauli, or *distance* of a stabilizer code must be at least three in order to correct one Pauli error on an n-qubit register. Several such codes exist [14, 81, 55, 77], along with analytic means of deriving them [82]. This suffices for reducing the logical error probability for the stochastic errors discussed above from order p_{err} to order p_{err}^2, given that a logical state is prepared, no operations (or noiseless operations) are carried out on the register, and then the state is measured. In the following sections, we examine methods for carrying out noisy operations on encoded registers so as to limit the effects of the noise, and an additional means of suppressing errors by successive encoding.

3.6 Fault-Tolerant Computation

If, in order to alter the state on an encoded register, it is first necessary to decode that register, then there is a finite amount of time which each qubit must spend unencoded. This negates the performance of any potential error-correcting scheme, since data will be left unprotected from error for a finite amount of time in each computational step. Therefore, it is necessary to perform operations directly on encoded data. In this section, we discuss methods to accomplish this, and constraints on these methods.

In the previous section, the central criterion for logical Paulis (membership in the set $N(S) \backslash S$) was introduced, as a constraint on Pauli operators. In addition, there exist non-Pauli unitary operators which can be said to normalize a stabilizer group S. Selecting a

member S_* of the stabilizer, and a stabilized state $|\psi\rangle$, we perform a brief calculation to illustrate the constraint on such unitaries:

$$U|\psi\rangle = US_*|\psi\rangle = (US_*U^\dagger)U|\psi\rangle\,. \tag{3.14}$$

In order for the operator U to normalize S, then, the set of operators $\{USU^\dagger\}$ must be equivalent to the stabilizer S. Since the stabilizer group is a subgroup of the Pauli group, we focus on the operators which map each Pauli P to another Pauli $P' = UPU^\dagger$. These are called Clifford operators, and while they are not sufficient for universal computation [2], there is a large set of important tasks which can be accomplished using Clifford operators.

The Clifford operators can be generated by multiplying a small set of operators, acting on individual qubits/pairs of qubits:

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad P = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix} \quad CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Table 3.4: Matrix representations of the primitive Clifford operators.

These operators can also be completely described by their action on generating sets of the Pauli group (typically X_j and Z_j for $0 \leq j \leq n-1$):

$$H = \begin{matrix} X \mapsto Z \\ Z \mapsto X \end{matrix} \quad P = \begin{matrix} X \mapsto Y \\ Z \mapsto Z \end{matrix} \quad CNOT = \begin{matrix} X_1 \mapsto X_1X_2 \\ X_2 \mapsto X_2 \\ Z_1 \mapsto Z_1 \\ Z_2 \mapsto Z_1Z_2 \end{matrix}$$

Table 3.5: Representations of the primitive Clifford operators by their action on the generators of the Pauli group $\{\, X_j,\, Z_j \mid j \in \mathbb{Z}_n \,\}$.

Clifford operators can be specified using a number of parameters which is quadratic in the number of qubits using the system above.

Note that the $CNOT$ can map weight-one Paulis to weight-two Paulis. For a stabilizer code which can correct weight-one errors, the $CNOT$ can transform correctable errors to non-correctable errors. In order to implement a Clifford operator fault-tolerantly, then, it is sufficient for the Clifford to be *transversal*, having no physical $CNOT$ gates (or other controlled-Pauli gates) on a given logical qubit. It is permissible to use physical $CNOT$

gates to implement two-qubit logical gates, as long as the control and target physical qubits are components of separate logical qubits, since a weight-one error on one code block will be mapped to two separate correctable errors.

The ability to fault-tolerantly apply Clifford gates to stabilizer states is insufficient for universal computation [2]. Furthermore, not all stabilizer codes admit a complete set of transversal Cliffords. However, some codes, such as the punctured Reed-Muller code on 15 qubits [84] admit transversal implementations of *non*-Clifford gates. This raises the question of whether a universal set of transversal gates can be obtained for a given code. Eastin and Knill [28] showed that no quantum code which detects a local error can possess a universal set of unitary gates, if all of the gates are transversal with respect to a fixed partition of the register.

In order to design protocols for universal fault-tolerant computation, it is necessary to circumvent this no-go result. The most popular means of doing so is gate teleportation, which uses controlled-Pauli gates and non-stabilizer states (called 'magic states') to implement non-Clifford gates [91]. Crucial to this approach is *magic state distillation*, the use of Clifford gates and projective measurements to extract small numbers of magic states from large registers initialized in noisy magic states [12]. Recently, Paetznick and Reichardt [62] showed that there exist codes which admit a transversal universal gate set on a subset of the protected qubits, provided that there are unused 'gauge' qubits prepared in a fixed state. These transversal operations induce Pauli errors on these gauge qubits which can be corrected using standard error-correction procedures.

3.6.1 Concatenation

In order to suppress errors to arbitrary precision, it is possible to concatenate a stabilizer code; each logical qubit being re-encoded into the same stabilizer code. We return to the three-qubit repetition code correcting bit-flip:

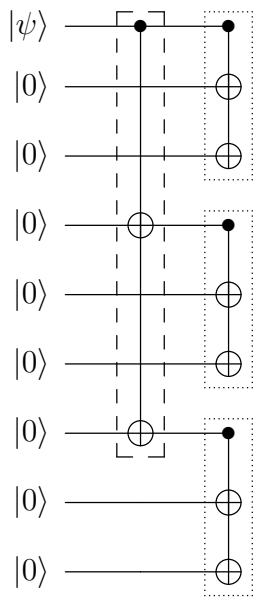


Figure 3.4: An encoding circuit for the correction of bit-flip errors using nine qubits. The state $|\psi\rangle$ is first encoded as in Figure 3.3 on qubits 1, 4, and 7 (dashed box). This encoding is then repeated on three registers, resulting in a second level of concatenation (dotted boxes).

The stabilizer for this code includes three sets of stabilizers for the three-bit repetition code, on neighbouring sets of three qubits, as well as one set of stabilizers expressed using the logical operators of the three-bit code:

$$S = \langle Z_1Z_2,\ Z_2Z_3,\ Z_4Z_5,\ Z_5Z_6,\ Z_7Z_8,\ Z_8Z_9,\ Z_1Z_2Z_3Z_4Z_5Z_6,\ Z_4Z_5Z_6Z_7Z_8Z_9\rangle\,. \quad (3.15)$$

This technique, applied to a code which can correct generic errors, produces an encoder with depth (number of timesteps required to execute a given circuit) ld, where l is the number of concatenation levels and d is the depth of the bottom-level encoder, and produces a code with a number of stabilizer generators $(n+1)^{l-1}r$, where n is the number of qubits used by the bottom-level code, and r is the number of stabilizer generators used by the bottom-level code. Note that, at the lowest level of concatenation, errors with probability of order p are replaced by errors with probability of order p^2. The second level of concatenation maps errors of order p^2 to errors of order p^4, and the l^{th} level produces an effective error of order p^{2^l}, which scales double-exponentially in l. In this way, only logarithmically-many levels of concatenation are required in order to ensure that the error rate decays exponentially in the size of the register. This analysis holds whenever the order-p^2 error probability is less

than the original error probability p. If $p \mapsto cp^2$ under concatenation, then $p < 1/c$ ensures that concatenation improves the error rate [52]. There are associated threshold error rates for the success of various fault-tolerant operations, see [3] for a comprehensive treatise.

3.7 Conclusion

Stabilizer codes based on the Pauli group are an extremely useful tool in quantum error correction. They provide a means of describing a quantum error-correcting code on n qubits, using only a quadratic number of parameters, which is a boon to analysis and simulation. Operations can also be performed directly on encoded data, in such a way that errors do not propagate within the registers associated with the same logical qubits. Assuming that the encoding operations are perfect, and error probabilities are sufficiently small, independent Pauli errors can be suppressed to arbitrarily low levels using concatenation, which does not consume a prohibitive amount of resources. In Chapter 5, we discuss improvements to common error-correcting codes, given that the ancilla states may not be pure before encoding. In Chapter 7, we discuss a potential alternative to the use of gate teleporation and magic states, which would use Clifford-based computation to alter the stabilizer code into which some data is encoded, without altering the data itself.

Chapter 4

Quantum Communication

Another important application of quantum information theory, including the theory of quantum codes, is the problem of transmitting *classical* information through noisy quantum channels. This differs from the central problem in quantum error correction in that it is not necessary for the quantum state itself to be preserved after transmission through the channel. In order to consider an attempt at classical communication through a quantum channel a success, it is only necessary to infer which state of a fixed set (typically called an *alphabet*) was sent. Compare this with quantum error correction, in which, in order for a procedure to be considered successful, a quantum state must be preserved independently of its membership in a given alphabet.

It is important to note that classical communication theory can be expressed in the language of quantum mechanics, with the restriction that states sent through channels are taken from an orthonormal alphabet, $\{|j\rangle\}$, $0 \leq j \leq d-1$ for some d-dimensional system, and that the channels probabilistically transform the alphabet states $|j\rangle \mapsto |j'\rangle$, without any accompanying unitary rotation, similarly to generalized amplitude damping, introduced in Equation 2.9. Another similarity between classical and quantum communication is that the amount of information which can be extracted from a state on a space A by measuring on another space B is given by the mutual information:

$$S(A:B) = S(\rho_A) + S(\rho_B) - S(\rho_{AB}), \tag{4.1}$$

which is zero for states $\rho_{AB} = \rho_A \otimes \rho_B$. In order to discover the properties of quantum information which deviate from intuition and classical theory, the focus in quantum information theory tends toward non-orthogonal alphabets [32], 'overloaded' alphabets which contain more than d states [49], and on non-standard channels, constructed with the goal of

exhibiting a certain property (See [80] for an example, though the cited work is concerned with a different information capacity than the one considered here).

In the first step of a quantum communication protocol, a transmitter (traditionally named Alice), selects a classical label corresponding to one of the states in the agreed-upon alphabet. She then prepares a quantum state corresponding to the classical label, and transmits it through a quantum channel. When Bob receives this quantum state, he performs a measurement, and attempts to deduce the original classical label. Since the protocol is considered successful when Bob's post-measurement classical state matches Alice's pre-preparation classical state, the fundamental problem under consideration is the transmission of classical information over quantum channels. The amount of information that can be transmitted using optimal preparations and measurements is quantified by the Holevo bound. A derivation for this quantity is presented in [88], it is presented below for completeness.

4.1 The Holevo Capacity

Suppose that Alice prepares a state $|\psi_j\rangle\langle\psi_j|$ with probability p_j from a known alphabet, also preparing (and subsequently retaining) a 'label' state $|j\rangle\langle j|$. The combined state of the system is a tensor product between Alice's Hilbert space A and the Hilbert space T, which is to be transmitted to Bob:

$$\rho_{AT} = \sum_j p_j \, |j\rangle\langle j| \otimes |\psi_j\rangle\langle\psi_j| \tag{4.2}$$

After transmission, the state is a tensor product between Alice's and Bob's respective Hilbert spaces:

$$\rho_{AB} = \sum_j p_j \, |j\rangle\langle j| \otimes \rho_j, \tag{4.3}$$

where $\rho_j = \Lambda(|\psi_j\rangle\langle\psi_j|)$.

There are two simple ways to formulate the Holevo quantity. The first is to consider the average change in entropy (see Chapter 1) when Bob identifies the index j. The state of Bob's system if the space A is traced out is $\sum_j p_j \rho_j$. This changes to ρ_j when Bob learns j, with the difference in entropy being $S(\rho_j) - S\left(\sum_j p_j \rho_j\right)$. Averaging over the distribution $\{\, p_j \,\}$ gives the Holevo quantity.

Alternatively, one can calculate the mutual information between Alice and Bob after transmission:

$$S(A:B) = S(A) + S(B) - S(A,B)$$
$$S(A) = S\left(\sum p_j \left|j\right\rangle\!\left\langle j\right|\right) = H(p)$$
$$S(B) = S\left(\sum p_j \rho_j\right)$$

The calculation of $S(A,B)$ is more complicated, and is given in [89]. It begins by taking the spectral decomposition of ρ_{AB},

$$\rho_{AB} = \sum_{j,k} p_j p_{k_j} \left|j\right\rangle\!\left\langle j\right| \otimes \left|k_j\right\rangle\!\left\langle k_j\right|,$$

and proceeds, using the spectral theorem to simplify the matrix logarithm:

$$\begin{aligned}
S(A,B) &= S\left(\sum_{j,k} p_j p_{k_j} \left|j\right\rangle\!\left\langle j\right| \otimes \left|k_j\right\rangle\!\left\langle k_j\right|\right) \\
&= -\mathrm{tr}\left(\left(\sum_{j,k} p_j p_{k_j} \left|j\right\rangle\!\left\langle j\right| \otimes \left|k_j\right\rangle\!\left\langle k_j\right|\right) \log\left(\sum_{j',k'} p_{j'} p_{k'_j} \left|j'\right\rangle\!\left\langle j'\right| \otimes \left|k'_j\right\rangle\!\left\langle k'_j\right|\right)\right) \\
&= -\sum_{j,j',k_j,k'_j} p_j p_{k_j} \log(p_{j'} p_{k'_j}) \mathrm{tr}\left(\left|j\right\rangle\!\left\langle j\right|j'\right\rangle\!\left\langle j'\right| \otimes \left|k_j\right\rangle\!\left\langle k_j\right|k'_j\right\rangle\!\left\langle k'_j\right|\right) \\
&= -\sum_{j,k} p_j p_{k_j} \log(p_j p_{k_j}) \\
&= -\sum_{j,k} p_j p_{k_j} \log(p_j) - \sum_{j,k} p_j p_{k_j} \log(p_{k_j}) \\
&= H(p) + \sum_j p_j S(\rho_j)
\end{aligned}$$

The mutual information between Alice and Bob after transmission, then, is equal to the Holevo quantity. This is the maximum amount of information that can be transmitted through a quantum channel *serially*, preparing one d-dimensional state at a time from the alphabet $\left|\psi_j\right\rangle\!\left\langle\psi_j\right|$ according to the discrete probability distribution $\{p_j\}$, and sending it through the channel Λ. In order to determine how much information can be sent through a channel in principle, it is first necessary to maximize over the sets $\{\left|\psi_j\right\rangle\!\left\langle\psi_j\right|\}$ and $\{p_j\}$:

$$C_H(\Lambda) = \max_{\{\left|\psi_j\right\rangle\!\left\langle\psi_j\right|\},\{p_j\}} S\left(\sum_j p_j \rho_j\right) - \sum_j p_j S(\rho_j) \tag{4.4}$$

This is the Holevo capacity [42], it is closely related to the classical capacity of a quantum channel. However, the two are not identical, because of the serial constraint mentioned earlier. Physically, it is possible for Alice to prepare an entangled state and send the subsystems through the channel Λ. To express the classical capacity, then, we must regularize the Holevo capacity:

$$C_{\text{classical}}(\Lambda) = \lim_{n\to\infty} \frac{1}{n} C_H(\Lambda^{\otimes n}). \tag{4.5}$$

If this regularized capacity is greater than the Holevo capacity, then the channel capacity is *super-additive*. In the next section, we discuss what is known with respect to super-additivity of the Holevo capacity.

4.2 Super-additivity

There are four related conjectures in quantum information theory, which were proven by Shor in 2003 [78] to be identical:

1. The Holevo capacity is additive.
2. The minimum output entropy $\left(S_{\min}(\Lambda) = \min_{|\psi\rangle\langle\psi|} \Lambda(|\psi\rangle\langle\psi|)\right)$ is additive.
3. The entanglement of formation is additive.
4. The entanglement of formation is also strongly super-additive.

(Since this section is concerned with the relation between conjectures 1 and 2, above, discussion of the entanglement of formation is omitted.)

The minimum output entropy is the simplest of these quantities, and it became the focus of study in this area. In 2009, Hastings [39] showed that there exist certain pairs of channels $\{\Lambda_1, \Lambda_2\}$ for which $S_{\min}(\Lambda_1 \otimes \Lambda_2) < S_{\min}(\Lambda_1) + S_{\min}(\Lambda_2)$. This implies that there exist channels or sets of channels for which $C_H\left(\bigotimes_j \Lambda_j\right) > \sum_j C_H\left(\Lambda_j\right)$.

However, no explicit example of the super-additivity of the Holevo capacity has been found. The applicability of the Hastings result to the search for such channels is not obvious, for two reasons, which were laid out by Fukuda, King and Moser in [33]. Firstly, the dimension of the space on which the Hastings result holds are lower-bounded by $7.8 \times 10^{32} \sim 1.2 \times 2^{109}$, requiring calculations on a 218-qubit register in order to see an explicit

example. This is numerically prohibitive. In addition, there is a lower bound of ~ 39000 on the rank of such a channel, further prohibiting direct numerical investigation. The specific channels studied by Hastings have also proven to possess additive Holevo capacities, by King in 2002 [48].

Recently, Belinschi, Collins and Nechita [8] showed that a macroscopic amount of sub-additivity, approaching one bit, can be attained in the minimum output entropy with spaces which are 183-dimensional, or between 7 and 8 qubits. It remains to be seen whether similar results can be obtained for the Holevo capacity, this will be discussed in Chapter 6.

Chapter 5

Quantum Error Correction with Mixed Ancilla Qubits

(This chapter is adapted from [20].)

This chapter is concerned with improving the encoding unitaries used in quantum error correction to better prevent the propagation of noise in the initial state of the ancillae, the $n-k$ qubits to which the system is coupled before encoding. In the simple scenarios which this document has considered, the implementation of an error-correcting code can be divided into four operations:

1. The one-qubit input state $\alpha\left|0\right\rangle+\beta\left|1\right\rangle$ is attached to an ancilla and a unitary operation rotates the state into the final encoded state $\alpha\left|\bar{0}\right\rangle+\beta\left|\bar{1}\right\rangle$, where $\{\left|\bar{0}\right\rangle,\left|\bar{1}\right\rangle\}$ are a set of orthogonal states in the larger Hilbert space.

2. Each qubit in the register is subjected to the random error process that the code is designed to correct.

3. The inverse of the encoding unitary is applied. The density matrix for the resulting state will contain terms proportional to $U\left|\psi\right\rangle\left\langle\psi\right|U^{\dagger}\otimes\left|s\right\rangle\left\langle s\right|$, where $\left|\psi\right\rangle$ is the original state and s is a classical $n-1$-bit string, the *syndrome* of the error U.

4. A unitary, controlled on the syndrome qubits, inverts the unitary in the terms described above, producing a final state which has greater fidelity to the input state than the state resulting from unencoded transmission.

It is useful to think of a quantum error correcting code as diverting entropy accrued during transmission to the ancilla qubits. It is often assumed that the qubits which comprise the ancilla are initialized in a pure state $|0\rangle$, or a state with negligible entropy. However, this assumption can only be satisfied approximately in practice. The difficulty in creating fiducial states at thermal equilibrium was discussed in Chapter 1; we can also consider the state of the ancilla immediately after either the procedure in Figure 3.3 or Figure 5.2 has been performed. This returns an ancilla state with higher entropy, which must be 'refreshed' to $|00\rangle$ in order for the code to be used again. This can be accomplished, for example, by projective measurement. If the operation which accomplishes this is imperfect, the ancilla will retain some of the entropy it gained during error correction.

In addition, many quantum computing architectures exist in which qubits equilibrate into Boltzmann distributions. Consider, for example, low-temperature solid-state ESR [34, 7, 79], where the relative population of the ground state of an electron spin is $\sim 3/4$ at 4.2 Kelvin and 7 Tesla. Throughout the remainder of this chapter, we treat the initial state as the result of a noisy process which occurs before the encoding operation.

In the following sections, we detail the error map that produces the ancilla noise we consider, and describe an additional operation to be performed prior to encoding (also known as an *augmentation* to an error-correcting code) which prevents some of the deleterious effects of this initialization error. We proceed to test this augmentation on two widely-studied error correction codes, correcting bit-flip and depolarization. We conclude by examining the effects of augmentation on a concatenated code.

5.0.1 Initialization Error

We assume each ancilla qubit is initialized in the state

$$\rho_q = \begin{bmatrix} 1 - q/2 & 0 \\ 0 & q/2 \end{bmatrix}. \tag{5.1}$$

In order to study the effect of initialization noise, we wish to model the noisy initial state as the result of an error process. There are many such processes that take the state $|0\rangle\langle 0|$ to ρ_q, such as the generalized amplitude damping presented in Chapter 2. A much simpler error process which accomplishes the same task is the bit-flip channel described below,

$$\Lambda = \left\{ \sqrt{1 - \frac{q}{2}}\hat{\mathbb{1}}, \sqrt{\frac{q}{2}}\hat{X} \right\}, \tag{5.2}$$

where the channel is given in the Kraus representation detailed in Chapter 2. The initial state on the ancillae is $\Lambda^{\otimes n}\left[|0\rangle\langle 0|^{\otimes n}\right]$. This additional error limits the ability of the ancilla to absorb entropy. As a result, any error-correcting code with ancilla qubits being maximally mixed, with $q = 1$, will not permit error syndromes to be measured for use in error correction. Note that maximally-mixed states *can* be used to encode information into decoherence-free subspaces [53] (see [57] for an introduction to decoherence-free subspaces). However, decoherence-free subspaces are not applicable to the error channels we examine in this chapter.

5.1 Augmented Error Correction

When subjected to initialization error, the controlled operations in the correction stage can introduce new errors into the output state, since the syndrome has been altered by the initialization error. A new code can be created to mitigate this error by implementing the inverse of the correction operation before the encoding unitary. This is shown in Figures 5.1 and 5.2.

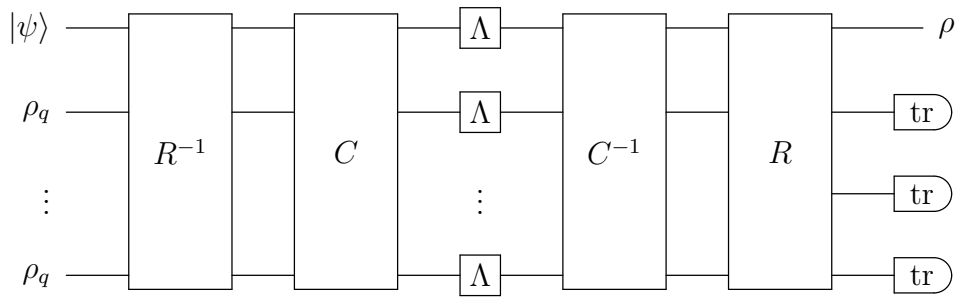


Figure 5.1: In order to augment an error correction code on n qubits, the recovery operator is inverted and implemented before encoding. This eliminates faults caused solely by false syndromes. Here, R is the recovery operator, C is the encoding operator and Λ is the error map.

The augmented three-qubit code in Figure 5.2 can be shown to satisfy a numerically-derived upper bound for the channel fidelity using an arbitrary CPTP/unital channel for encoding. To derive this bound, the optimization of channel fidelity is posed as a semi-definite program [68], optimizing over the encoding channel, which is linearly constrained

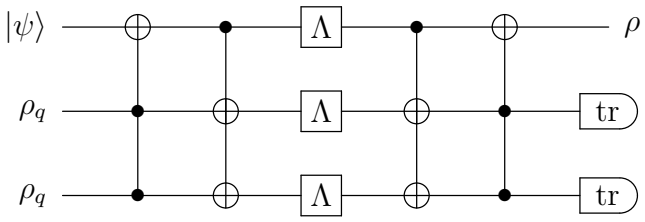


Figure 5.2: The traditional 3-qubit code to correct bit-flip errors (seen in Chapter 3), augmented to provide increased fidelity, in the case where each ancilla qubit is subject to the initialization noise map discussed above. The map E in this example is the bit-flip map $\left\{\sqrt{1-p}\hat{\mathbb{1}}, \sqrt{p}\hat{X}\right\}$. The augmentation consists of implementing the Toffoli used to correct detected errors before the standard encoding procedure takes place. This improves the overall fidelity by ensuring that, if no error occurs, the encoded state remains unaltered by false syndromes.

to be both CPTP and unital. The numerical search for optimal encoders is the origin of the augmentation in this chapter. For more information on this numerical method, see Appendix A.

The advantages of this augmentation are:

1. The fidelity of the augmented codes will always exceed or equal that of the unaugmented codes, since the augmentation corrects additional errors left uncorrected by the unaugmented codes without altering the function of the error correcting code for pure ancillae.

2. The augmented code is especially useful in implementations where the main error parameter p can be constrained, and the initialization parameter q cannot. For example, when the error parameter during a storage operation is time-dependent, reducing the storage time reduces the error parameter. This is true for any error channel, and any number of ancilla qubits, since the inverse recovery operator prevents faults in the case where the error map acts trivially. Therefore, when p can be diminished to arbitrary size, the augmented code allows arbitrarily high fidelity, where the unaugmented code does not.

3. Augmented codes provide increased fidelity at higher q than unaugmented codes. A code (whose implementation will be denoted Θ) is *useful* if, for an error channel Λ, $F_C(\Theta) \geq F_C(\Lambda)$.To illustrate this, we plot the tolerable q in Figures 5.3, 5.4 and 5.5.

	# Qubits	c_0	c_1
Unaugmented	3	$1 - {}^1\!/_4 q^2$	$-2q + {}^3\!/_2 q^2$
	5	$1 - {}^1\!/_2 q^3 + \ldots$	$-{}^9\!/_2 q^2 + 6q^3 - \ldots$
	7	$1 - \frac{15}{16} q^4 + \ldots$	$-10q^3 + \ldots$
	9	$1 - {}^7\!/_4 q^5 + \ldots$	$-{}^{175}\!/_8 q^4 - \ldots$
Augmented	3	1	$-2q + {}^1\!/_2 q^2$
	5	1	$-{}^9\!/_2 q^2 + 3q^3 + \ldots$
	7	1	$-{}^5\!/_{16} q^3 + \ldots$
	9	1	$-{}^{175}\!/_8 q^4 - \ldots$

Table 5.1: Fidelity coefficients for four repetition codes, correcting bit flip. Each fidelity is expressed as a polynomial in p, $F_C = \sum_k c_k p^k$, the c_0, c_1 are shown. Note that, for the augmented codes, $c_0 = 1$, indicating that the contribution to the error term due solely to the mixed ancilla has been eliminated.

Furthermore, note that this procedure increases the gate complexity of the code by at most a factor of 2, since the augmenting unitary is already required for the code to function. We conclude that this augmentation will be useful in a variety of circumstances, and in the following sections, we examine examples of this strategy used to counter two common error processes; bit flip and depolarization.

5.2 Bit Flip

In order to correct Pauli-$\hat{X}$ (bit flip) errors, we encode the state we wish to preserve into the 2-dimensional subspace of an n-qubit (2^n-dimensional) register having maximum distinguishability under bit flip; $\{|0\rangle^{\otimes n}, |1\rangle^{\otimes n}\}$. In order to correct t^{th}-order bit flip errors, $2t + 1$ qubits are required. Here, we analyse 3-, 5-, 7- and 9-qubit repetition codes to counter bit flip errors, with and without augmentation. Each fidelity is expressed as a polynomial in p, $F_C = \sum_k c_k p^k$, the c_0, c_1 are shown in Table 5.1.

We conclude by noting that this behaviour can be trivially extended to codes that correct any channel of the form $\left\{\sqrt{1-p}\hat{\mathbb{1}}, \sqrt{p}UXU^\dagger\right\}$.

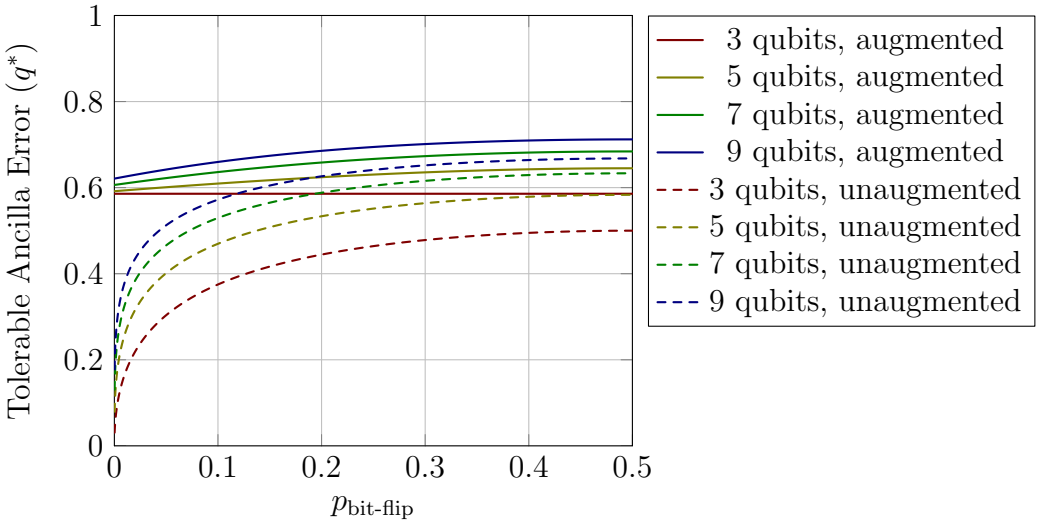


Figure 5.3: The initialization error for which an error correcting code can give a channel fidelity $\geq 1 - p$. This is shown for four repetition codes correcting bit-flip errors (3, 5, 7, and 9 qubits, shown in red, yellow, green and blue, respectively). Note that the tolerable error for small values of p, the parameter describing the main bit-flip channel, approaches 0 rapidly for unaugmented codes (dashed lines). By contrast, augmentation (solid lines) provides a high tolerable q for every value of p.

5.3 Depolarization

It is important, in order to ensure that augmented error correction codes are widely useful, to examine the performance of such codes correcting depolarization, an error process to which all error processes can be reduced [21]. Depolarization is a channel which consists of the following Kraus map:

$$\Gamma = \left\{ \sqrt{1 - \frac{3p}{4}}\hat{\mathbb{1}}, \sqrt{\frac{p}{4}}\hat{X}, \sqrt{\frac{p}{4}}\hat{Y}, \sqrt{\frac{p}{4}}\hat{Z} \right\} \tag{5.3}$$

We find channel fidelities for an augmented 5-qubit code versus depolarization, and an unaugmented code [55, 9].

Here, the optimization of the channel fidelity has not been posed as a semi-definite problem. Instead, we have assumed that a unitary will be appended to the encoder which consists of 2^{n-1} single-qubit unitaries, each controlled on a unique binary string on the

	c_0	c_1
Unaugmented	$1 - {}^3/_2 q^2 + q^3 - \ldots$	$-6q + {}^{21}/_2 q^2 - {}^{11}/_2 q^3 + \ldots$
Augmented	1	$-6q + {}^9/_2 q^2 - {}^3/_2 q^3 + \ldots$

Table 5.2: Fidelity coefficients for the 5-qubit perfect code, correcting depolarization. Each fidelity is expressed as a polynomial in p, $F_C = \sum_k c_k p^k$, the c_0, c_1 are shown. Coefficients for the unaugmented code are above, those for the augmented code below. Note that, for the augmented codes, $c_0 = 1$, indicating that the contribution to the error term due solely to the mixed ancilla has been eliminated.

ancilla. This reduces the size of the optimization problem from 4^n to $3 \cdot 2^{n-1}$, each single-qubit unitary having 3 free parameters. We observe that the optimal unitary is the inverse of the correcting operation.

We present the polynomial coefficients for the fidelity, as described in Table 5.2. Here, we see that the p-independent terms are eliminated, but the term linear in p remains. We continue, showing the tolerable initialization noise levels for codes that counter depolarization errors in Figure 5.4

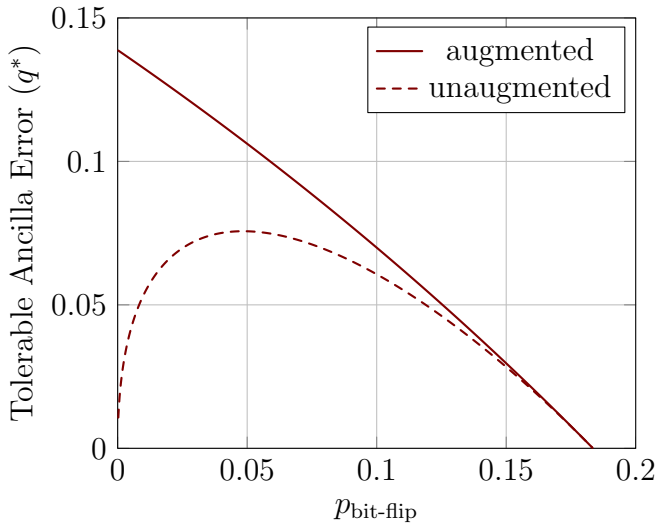


Figure 5.4: The initialization error for which an error correcting code can give a channel fidelity $\geq 1 - \frac{3}{4}p$. This is shown for the perfect 5-qubit code. Note that the behaviour of this code is qualitatively different, having 0 tolerable initialization for $p \sim 0.18$. The ability of the augmented code to provide finite tolerable q at $p = 0$ is preserved.

	c_0	c_1
Unaugmented	$1 - 1/4q^2 + 1/2q^3 + \ldots$	$-4q^2 + 3q^3 + \ldots$
Top-Level Augmented	$1 - 1/2q^3 + \ldots$	$-4q^2 + q^3 + \ldots$
Fully Augmented	1	$-4q^2 + 2q^3 + \ldots$

Table 5.3: Fidelity coefficients for the two-level concatenated repetition code, correcting bit flip. Each fidelity is expressed as a polynomial in p, $F_C = \sum_k c_k p^k$, the c_0, c_1 are shown. The unaugmented code is presented, first, followed by the top-level augmented code and the fully-augmented code.

5.4 Concatenation

It is useful to examine the effect of augmentation on a two-level concatenated code, in order to determine the benefits of augmentation at each level. Below, we examine the effect of augmentation on the concatenated 3-qubit code. With bit-flip probability p and initialization error q as defined above, the channel fidelity for unaugmented, top-level augmented, and fully augmented codes are shown in Table 5.3 The tolerable initialization noise is shown in Figure 5.5.

5.5 Discussion & Summary

The large initialization errors discussed in this chapter render fault-tolerant computation impossible with current methods. The purpose of the augmented error correction described above is to partially compensate for these errors, and to increase the utility of highly mixed states. This technique is intended for experimental use in the near term, in venues such as solid-state nuclear magnetic resonance (SSNMR), which does not possess an easy means of refreshing ancilla qubits, and where the error introduced by implementing the additional recovery operator is likely to be much smaller than the error in ancilla state preparation. An emphasis has been placed on avoiding the incorporation of additional ancilla qubits, since experimental implementation is currently restricted to small registers. This is true not only for SSNMR, but in other venues as well.

The recovery operator in a stabilizer error correcting code, such as those shown above, is costly to implement fault-tolerantly. This is due to the fact that Pauli gates which are controlled on $n-1$ qubits are not in the Clifford group, a set of gates that can be implemented without causing adverse error propagation in deeply concatenated error correcting codes. This has motivated the development of alternate recovery procedures, such as those

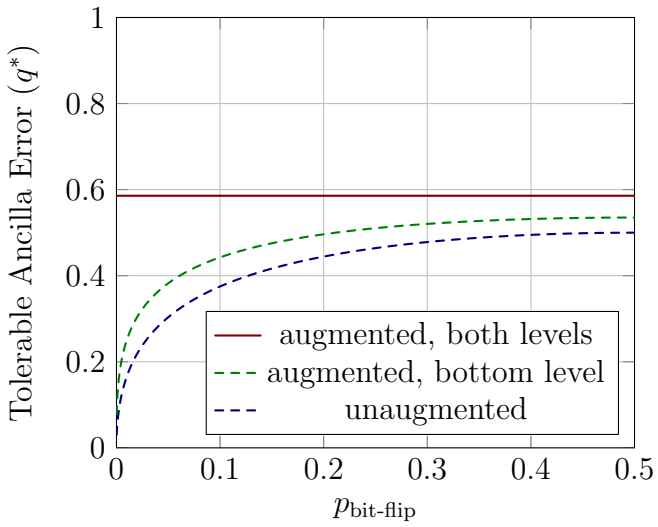


Figure 5.5: The tolerable initialization noise for the twice-concatenated 3-qubit code correcting bit-flip. For the top-level concatenation, the encoder used in the top of Figure 5.2 has been replaced with the encoder used in the bottom of Figure 5.2. For the full concatenation, all the encoding circuits used are augmented, as in Figure 5.2. Note that the bottom curve (for the unaugmented concatenated code) is identical to the tolerable initialization noise for the 3-bit error correcting code when left unconcatenated, shown in Figure 5.3. Also, the tolerable q for the fully augmented code is $2 - \sqrt{2}$, identical to the augmented 3-qubit code.

used in Knill error correction [50]. It is possible that a fault-tolerant recovery scheme can be adapted to the task of preventing ancilla error propagation. It remains to be seen, however, whether this will increase the error threshold for fault-tolerant protocols.

In summary, the assumption that there exists a pure ancilla, initialized in the state $|0\rangle^{\otimes n}$ is often violated, since the initialization process is imperfect in practice. This motivates the study of error correcting codes whose encoding operators are augmented to produce higher fidelities in the presence of initialization errors. The augmentation consists of inverting the recovery operator (which performs a single-qubit unitary on the message qubit, controlled on the end state of the ancilla qubits) and inserting it before encoding. The action of this augmented encoding can be easily understood from Figure 5.1; it ensures that, if the main error channel acts trivially, the output state is equal to the input state, as opposed to having been altered by the false syndrome generated by the initialization noise. This augmentation produces fidelities strictly greater than those from unaugmented codes, and

constrains all error terms to be proportional to the main error probability, useful when that parameter can be controlled experimentally.

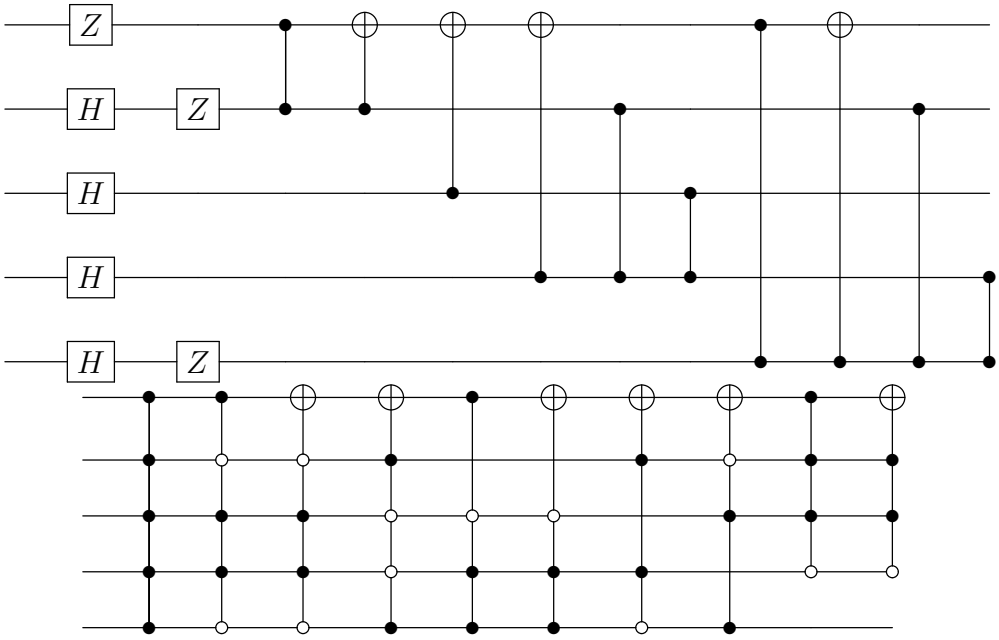


Figure 5.6: The augmented version of the 'perfect' 5-qubit code given in [36] and errata. The circuit above is the unaugmented encoder, the circuit below is the correction operator. These are combined according to the prescription in Figure 5.1. Here, the error channel is the depolarizing channel $\{\sqrt{1-3p/4}\hat{\mathbb{1}}, \sqrt{p/4}\hat{X}, \sqrt{p/4}\hat{Y}, \sqrt{p/4}\hat{Z}\}$. The augmentation has a similar effect to that used on the $(2t+1)$-qubit codes countering t^{th}-order bit flip errors, shown in Figure 5.3. We can deduce from this that the benefits of augmentation as described above are not limited to codes which counter classical errors.

Chapter 6

Search for Concrete Examples of Super-Additivity of the Holevo Capacity

This Chapter is intended to discuss an in-depth numerical examination of the potential super-additivity of the Holevo capacity of a given channel, introduced in Chapter 4. The overall goal of this work is to provide a concrete demonstration of super-additivity, i.e. a numerical channel Φ and an input state $|\psi\rangle$ which is a member of a family of optimal inputs producing $C_H(\Phi^{\otimes n}) > nC_H(\Phi)$. We begin in Section 1, by defining the Holevo capacity both operationally, and using a formula which simplifies numerical evaluation. In Section 2, we discuss the means by which this capacity will be calculated, focusing on an iterative optimization algorithm developed by Shor (unpublished, though details of the algorithm's function can be found in [22]). In Section 3, we introduce the special class of channels for which we suspect super-additivity of the Holevo capacity will be apparent. In Section 4, we give the criteria we will use to determine whether a channel exhibits super-additivity. In Section 5, we present the results of numerical calculation, and we conclude in Section 6, presenting a series of directions for further research.

6.1 The Holevo Capacity

Recalling Chapter 4, the Holevo capacity is defined as the amount of classical information which can be extracted from a quantum communication channel using an optimal measurement, when inputs to the channel are restricted to separable states on a composite

quantum system [42, 73]. Since the inputs must be separable, they can be described on a single instance of the system of interest. We define an *ensemble* to be a set of (for the purposes of this chapter, single-qubit) states $\{ \rho_j \}$ with associated probabilities $\{ \pi_j \}$. The Holevo capacity is:

$$C_{\text{Holv}}(\Phi) = \sup_{\{ \pi_j, \rho_j \}} \chi_{\text{Holv}}(\Phi, \{ \pi_j, \rho_j \}) \tag{6.1}$$

$$\chi_{\text{Holv}}(\Phi, \{ \pi_j, \rho_j \}) = \left[S\left(\sum_j \pi_j \Phi(\rho_j) \right) - \sum_j \pi_j S(\Phi(\rho_j)) \right]$$

$$\text{where } S(\rho) = -\text{tr}\rho \log_2(\rho)$$

This definition is operational, being the maximum of the average distinguishability of the set of output states $\{ \Phi(\rho_j) \}$ from the 'average' output $\sum_j \pi_j \Phi(\rho_j)$. It is worthwhile to note that, since the action of Φ is linear, the average output from the channel is simply the action of Φ on the average input to the channel, $\sum_j \pi_j \rho_j$.

In order to evaluate the classical capacity of a quantum channel, it is necessary to *regularize* the Holevo capacity:

$$C_{\text{Classical}}(\Phi) = \lim_{n \to \infty} \frac{1}{n} C_{\text{Holv}}(\Phi^{\otimes n}) \tag{6.2}$$

If the Holevo capacity is super-additive, then the Holevo capacity of $\Phi^{\otimes n}$ is greater than $nC_{\text{Holv}}(\Phi)$ for some value of n. However, if the Holevo capacity is additive for a given channel, then regularization is not necessary, since $nC_{\text{Holv}}(\Phi) = C_{\text{Holv}}(\Phi^{\otimes n})$. In the following section, we discuss the process by which we determine empirically whether this capacity is additive for a given channel.

Equation 6.1 is operationally-defined, but prohibitively difficult to optimize numerically, since it requires simultaneous maximization over ensembles consisting of at most 4^n 2^n-dimensional vectors, each with an associated probability. Instead, we use a formula for the Holevo capacity which is defined in terms of the relative Von Neumann entropy between individual outputs from the channel and the output average [73, 61]:

$$C_H(\Phi^{\otimes n}) = \min_{\gamma} \max_{\psi} H\left[\Phi^{\otimes n}(|\psi\rangle\langle\psi|), \Phi^{\otimes n}(\gamma) \right] \tag{6.3}$$

$$\text{where } H[\rho, \sigma] = \text{tr}(\rho \log \rho - \rho \log \sigma),$$

where $|\psi\rangle$ is a single pure state, and γ is a density matrix. This expression for the Holevo capacity is related to Equation 6.1, in that the optimal γ is equal to the average of optimal inputs to the channel $\Phi^{\otimes n}$, and the optimal $|\psi\rangle$ is one of the optimal inputs to the channel. Since only a single 2^n-dimensional vector is required for the inner maximization, this maximization requires far fewer parameters than the optimization in Equation 6.1. We will argue that the inner maximization is sufficient to determine whether the Holevo capacity is super-additive for a given channel, reducing the number of parameters required in a numerical optimization.

To make this argument, we introduce a pseudo-capacity for Φ, using the tensor product of the single-qubit average optimal output as a substitute for γ:

$$\tilde{C}_H = \max_{\psi} H\left[\Phi^{\otimes n}(|\psi\rangle\langle\psi|), \Phi^{\otimes n}(\rho_{\text{av}}^{\otimes n})\right] \tag{6.4}$$

where ρ_{av} is the optimal input average for the one-qubit channel. One of two arguments holds. If $\rho_{\text{av}}^{\otimes n}$ is the minimum-satisfying value of γ, the pseudo-capacity is equal to the Holevo capacity. If $\rho_{\text{av}}^{\otimes n}$ is not the minimum-satisfying value of γ, then the optimal ensemble must contain states which are not tensor products of the states in the one-qubit optimal ensemble. This also indicates that the Holevo capacity is super-additive. In either case, we can learn whether the Holevo capacity is super-additive for a given channel simply by performing this inner maximization. The sufficiency of the inner maximization reduces the nested optimization over 4^n 2^n-dimensional vectors from Equation 6.1 to a simple optimization over a single 2^n-dimensional input state.

Note that it is sufficient to consider pure input states, rather than mixed states. This is because, with γ fixed at the value $\rho_{\text{av}}^{\otimes n}$, the relative entropy is convex in $|\psi\rangle$, and the mixed states are convex combinations of the pure states. Since the maximum of a convex function on a convex set lies on the boundary, it suffices to optimize over pure states. We detail, in the following section, a pair of algorithms for optimizing over pure states.

6.2 Methods For Calculating Capacity

In order to evaluate the Holevo capacity, it is necessary to optimize over 2^n-dimensional states. We detail how this optimization can be carried out, using a convergent update rule from Shor (unpublished, for proof of its convergence, see Appendix A of [22]). We also introduce a more efficient means of evaluating the action of a quantum channel of the form

$\Phi^{\otimes n}$ on an n-qubit density matrix, which is, if performed naïvely, the most numerically expensive component of the calculation (due to the technical nature of this discussion, we will only provide a brief introduction to the channel-application algorithm in this Chapter; for a comprehensive discussion, see Appendix B).

6.2.1 Shor Optimization

The relative Von Neumann entropy between $\Phi^{\otimes n}(|\psi\rangle\langle\psi|)$ and $\Phi^{\otimes n}(\rho_{\text{av}}^{\otimes n})$, used in Equation 6.3 to express the Holevo capacity, can be expressed as a trace whose cyclic property can then be exploited:

$$\begin{aligned} & H\left[\Phi^{\otimes n}(|\psi\rangle\langle\psi|), \Phi^{\otimes n}(\rho_{\text{av}}^{\otimes n})\right] \\ &= \operatorname{tr}\left(\Phi^{\otimes n}(|\psi\rangle\langle\psi|)\left(\log \Phi^{\otimes n}(|\psi\rangle\langle\psi| - \log \Phi^{\otimes n}(\rho_{\text{av}}^{\otimes n})\right)\right) \\ &= \langle\psi| \Phi_*^{\otimes n}\left(\log \Phi^{\otimes n}(|\psi\rangle\langle\psi| - \log \Phi^{\otimes n}(\rho_{\text{av}}^{\otimes n})\right)|\psi\rangle \equiv \langle\psi|\zeta(\psi)|\psi\rangle\,, \end{aligned} \tag{6.5}$$

where Φ_* is the *dual* channel to Φ, with respect to the Hilbert-Schmidt inner product $\operatorname{tr}(A^\dagger B)$. That is, $\operatorname{tr}(\Phi_*(A)^\dagger B) = \operatorname{tr}(A^\dagger \Phi(B))$. The action of a dual channel is roughly as expensive to compute as the action of the corresponding channel. For example, if the action of Φ in the Kraus representation is $\sum_j F_j \rho F_j^\dagger$ the action of Φ_* will be $\sum_j F_j^\dagger \rho F_j$. The matrix element in Equation 6.5 is maximized when $|\psi\rangle$ is the principal eigenvector of $\zeta(\psi)$. Also, for any $|\psi\rangle$, the principal eigenvalue of $\zeta(\psi)$ is greater than or equal to $\langle\psi|\zeta(\psi)|\psi\rangle$. Therefore, if we begin with a random state $|\psi\rangle$, and iteratively update $|\psi_n\rangle$ to the principal eigenvector of $\zeta(\psi_{n-1})$, we obtain convergence to a value of $|\psi\rangle$ which is locally maximal.

In order to calculate the result of this update rule, it is necessary to calculate the action of Φ and Φ_* on large matrices. If implemented naïvely, this is the most computationally intensive of the operations that must be performed. In the following section, we detail a way to take advantage of the tensor product structure of the channel $\Phi^{\otimes n}$ to evaluate its action more efficiently than can be done for general n-qubit channels.

6.2.2 Evaluation of Channel Action

To show how channels of the form $\Phi^{\otimes n}$ can be efficiently implemented, we first establish a case for comparison, the evaluation of channel action for general channels. As discussed in

Chapter 2, in order to evaluate the action of an n-qubit channel on a state ρ, 2^{4n} operations are required for super-matrix based channel representations, 2^{5n} in the Kraus representation if the channel possesses 2^{2n} Kraus operators (the maximum number necessary to describe a channel on n qubits). Evaluation of channel action, therefore, is the most costly step in calculating the Holevo capacity, since the numerical evaluation of the matrix logarithm and the principal eigenvector require a number of operations on the order of 2^{3n} [35].

Fortunately, it is possible to take advantage of the structure of $\Phi^{\otimes n}$ to perform the required evaluation in an amount of time which requires order $n \cdot 2^{2n}$ operations, requiring only the original Φ (a 1-qubit channel) to be stored. In brief, the reduction in the number of operations is due to two special properties of the channel. The first is that a channel of the form $\Phi^{\otimes n}$ can be expressed as a composition of n much simpler channels:

$$\Phi^{\otimes n} = \bigcirc_{j=1}^{n} \Phi_j; \quad \Phi_j = \hat{\mathbb{1}}^{\otimes j-1} \otimes \Phi \otimes \hat{\mathbb{1}}^{\otimes n-j}, \tag{6.6}$$

It is therefore sufficient to evaluate the action of Φ_j for all values of j from 1 to n. Each such evaluation requires an amount of work scaling as 2^{3n} in the Kraus representation, since the channels Φ_j each possess a maximum of 4 Kraus operators. To evaluate the action of these channels in series, then, requires an amount of work scaling as $n2^{3n}$. This can be further reduced to $n2^{2n}$, using a more advanced method detailed in Appendix B. With these efficient algorithms in hand, we advance, in the next section, to the task of selecting a channel which we believe will exhibit concrete super-additivity.

6.3 Channel Selection

We must select a channel Φ, for which we suspect $C_{\text{Holv}}(\Phi^{\otimes n}) > nC_{\text{Holv}}(\Phi)$, for some n. We examine a family of channels which have simple representations in the Pauli basis:

$$\begin{aligned} \rho = [\rho_x, \rho_y, \rho_z]\,,\ \rho_a = \text{tr}(\rho A) \\ [\rho_x, \rho_y, \rho_z] \mapsto [\lambda_x \rho_x, \lambda_y \rho_y, \lambda_z \rho_z + t_z]\,,\ 0 \leq \lambda_x, \lambda_y, \lambda_z, t_z \leq 1. \end{aligned} \tag{6.7}$$

These four parameters describe a reasonably general family of quantum maps (see Chapter 2). A pair of parameters $\{ t_x, t_y \}$, if included, would provide a completely general

description of any CPTP map up to a change of basis [47]. However, they are not necessary to derive a set of channels which we suspect will exhibit super-additivity, so they are set to 0 for simplicity. We discuss below the necessary constraints on the parameters $\lambda_x, \lambda_y, \lambda_z$, and t_z, as well as constraints which will limit the search to channels which have not been proven to possess additive Holevo capacities.

Any channel Φ must, first of all, be completely-positive and trace preserving (CPTP). Ruskai, Szarek and Werner [70] proved that, for the channel family given in Equation 6.7, the CPTP condition can be reduced to the Algoet-Fujiwara conditions:

$$(\lambda_x \pm \lambda_y)^2 \leq (1 \pm \lambda_z)^2 - t_z^2 \tag{6.8}$$

In addition, we note that there are two families of channels whose Holevo capacities have been proven additive. These are the unital channels [48], for which $\Phi\left(\hat{\mathbb{1}}/2\right) = \hat{\mathbb{1}}/2$, and the entanglement-breaking channels, for which all output states are separable [76]. Each of these constraints can be reduced to a constraint on the parameters $\{\, \lambda_x, \lambda_y, \lambda_z, t_z \,\}$.

In order to ensure that the channel we select is non-unital, we note that $\Phi\left(\hat{\mathbb{1}}/2\right) = \left(\hat{\mathbb{1}}/2\right) + t_z\left(Z/2\right)$, so $t_z \neq 0$ is a necessary condition for super-additivity. We will restrict t_z to be greater than 0, since channels given by Equation 6.7 which differ only in the sign of t_z are equivalent under a change of basis, and therefore have the same Holevo capacity.

In order to ensure that a channel preserves entanglement, it is necessary and sufficient [69] to show that its Choi state $\hat{\mathbb{1}} \otimes \Phi\left(|\Omega\rangle\langle\Omega|\right)$ (where $|\Omega\rangle = 1/\sqrt{2}(|00\rangle + |11\rangle)$) is entangled. This can be accomplished using the negativity of the partial transpose [63, 43], since the system in question is 2×2. The Choi state for a four-parameter channel is:

$$\hat{\mathbb{1}} \otimes \Phi\left(|\Omega\rangle\langle\Omega|\right) = \frac{1}{2}\begin{bmatrix} 1 + t_z + \lambda_z & 0 & 0 & \lambda_x + \lambda_y \\ 0 & 1 - t_z - \lambda_z & \lambda_x - \lambda_y & 0 \\ 0 & \lambda_x - \lambda_y & 1 + t_z - \lambda_z & 0 \\ \lambda_x + \lambda_y & 0 & 0 & 1 - t_z + \lambda_z \end{bmatrix} \tag{6.9}$$

The partial transpose acting on this matrix yields the same result as the transformation $\lambda_y \mapsto -\lambda_y$. Therefore, in order for the channel to preserve entanglement, one of the following conditions must hold:

$$(\lambda_x \mp \lambda_y)^2 > (1 \pm \lambda_z)^2 - t_z^2.$$

One of the entanglement-preserving constraints is not physical, since channels satisfying it cannot be CPTP. Consider the inequalities which contain the term $(1+\lambda_z)^2$:

$$\begin{aligned}\text{(CPTP)}&: (\lambda_x+\lambda_y)^2 \leq (1+\lambda_z)^2 - t_z^2\\ \text{(non-EB)}&: (\lambda_x-\lambda_y)^2 > (1+\lambda_z)^2 - t_z^2\end{aligned}$$

This implies that $(\lambda_x+\lambda_y)^2 < (\lambda_x-\lambda_y)^2$, which is impossible. We therefore reject the non-physical entanglement-preserving criterion $(\lambda_x-\lambda_y)^2 \leq (1+\lambda_z)^2 - t_z^2$, and use only $(\lambda_x+\lambda_y)^2 > (1-\lambda_z)^2 - t_z^2$, which can be satisfied for CPTP channels.

These constraints alone are not sufficient to specify a single channel Φ for further study. To further narrow the search, we further restrict the channel to have a property first described in [49], the necessity for an ensemble containing more than two states to be used in order to achieve the Holevo capacity. This channel property is non-classical, in that the capacity of a classical channel can always be attained with a two-state ensemble. It is hoped that there exists a three-state channel which exhibits super-additivity, since this is also a non-classical property.

In order to determine a three-state channel, we can use the sufficient conditions from [49]. We first introduce, for channels of the form given in Equation 6.7, two restricted capacities, the vertical and horizontal capacities:

$$\mathcal{E}_{\text{vert}} = \{(p,[0,0,1]),(1-p,[0,0,-1])\} \tag{6.10}$$

$$\begin{aligned}C_{\text{vert}}(\lambda_z,\, t_z) = \max_p\, & S([0,0,\lambda_z(2p-1)+t_z])\\ & - pS([0,0,\lambda_z+t_z]) - (1-p)S([0,0,-\lambda_z+t_z])\end{aligned} \tag{6.11}$$

$$\mathcal{E}_{\text{horz}} = \left\{(1/2,[\sqrt{1-z^2},0,z]),(1/2,[-\sqrt{1-z^2},0,z])\right\} \tag{6.12}$$

$$C_{\text{horz}}(\lambda_x,\, \lambda_z,\, t_z) = \max_z S([0,0,\lambda_z z+t_z]) - S([\lambda_x\sqrt{1-z^2},0,\lambda_z z+t_z]) \tag{6.13}$$

When these restricted capacities are equal ($C_{\text{vert}} = C_{\text{horz}}$), but the averages over the respective ensembles are not equal ($\rho_{\text{av(vert)}} \neq \rho_{\text{av(horz)}}$), then

$$\chi\left(\frac{1}{2}(\mathcal{E}_{\text{vert}}+\mathcal{E}_{\text{horz}})\right)=$$
$$\frac{1}{2}(C_{\text{vert}}+C_{\text{horz}})+S\left(\frac{1}{2}\Phi\left(\rho_{\text{av(vert)}}+\rho_{\text{av(horz)}}\right)\right)-\frac{1}{2}S\left(\Phi\left(\rho_{\text{av(vert)}}\right)\right)-\frac{1}{2}S\left(\Phi\left(\rho_{\text{av(horz)}}\right)\right)$$
$$>C_{\text{vert}}, \tag{6.14}$$

because the entropy is strictly concave. This implies that an ensemble of two states is insufficient to achieve the Holevo capacity of these channels. It has been shown that, for any qubit channel, an ensemble of four states is sufficient to achieve the Holevo capacity [23].

In principle, the constraint $C_{\text{vert}}=C_{\text{horz}}$ can be used to constrain λ_x for given values of λ_z and t_z. We will show below that this constraint can only be satisfied numerically, though C_{vert} can be found analytically.

The vertical capacity can be obtained by maximizing

$$\chi_{\text{vert}}=S([0,0,\lambda_z(2p-1)+t_z])-pS([0,0,\lambda_z+t_z])-(1-p)S([0,0,-\lambda_z+t_z])$$

To differentiate this, we introduce $z_{\text{out}}=\lambda_z(2p-1)+t_z$:

$$\chi_{\text{vert}}=S([0,0,z_{\text{out}}])-pS([0,0,\lambda_z+t_z])-(1-p)S([0,0,-\lambda_z+t_z])$$
$$\frac{\partial}{\partial p}\chi_{\text{vert}}=\lambda_z\log_2\left(\frac{1-z_{\text{out}}}{1+z_{\text{out}}}\right)-S([0,0,\lambda_z+t_z])+S([0,0,-\lambda_z+t_z])$$
$$\frac{\partial^2}{\partial p^2}\chi_{\text{vert}}=\frac{2\lambda_z}{z_{\text{out}}^2-1}$$

The second derivative is negative for $-1\leq z_{\text{out}}\leq 1$, so $\dfrac{\partial}{\partial p}\chi_{\text{vert}}=0$ indicates the unique maximum. Continuing the derivation:

$$\log_2\left(\frac{1-z_{\text{out}}}{1+z_{\text{out}}}\right)=\frac{S([0,0,\lambda_z+t_z])-S([0,0,-\lambda_z+t_z])}{\lambda_z}$$
$$z_{\text{out}}^*=\frac{1-k}{1+k};\quad k=2^{\frac{1}{\lambda_z}(S([0,0,\lambda_z+t_z])-S([0,0,-\lambda_z+t_z]))},\quad p^*=\frac{z_{\text{out}}^*-t_z+\lambda_z}{2\lambda_z}$$

This provides an analytic definition for C_{vert}, specifying the value of p for which the capacity is obtained (denoted p^* above). The corresponding derivation for $\dfrac{\partial \chi_{\text{horz}}}{\partial z}$ is as follows:

$$\frac{\partial \chi_{\text{horz}}}{\partial z} = \frac{\lambda_z}{2} \log\left(\frac{1 - z_{\text{out}}}{1 + z_{\text{out}}}\right) + \frac{\lambda_x^2 z - \lambda_z z_{\text{out}}}{2\sqrt{z_{\text{out}}^2 + \lambda_x^2(1 - z^2)}} \log\left(\frac{1 - \sqrt{z_{\text{out}}^2 + \lambda_x^2(1 - z^2)}}{1 + \sqrt{z_{\text{out}}^2 + \lambda_x^2(1 - z^2)}}\right) \tag{6.15}$$

Setting this quantity equal to 0 results in a transcendental equation, so in order to constrain λ_x for a given channel, it is necessary to ensure that this constraint is satisfied numerically. It is also necessary, therefore, to evaluate $\rho_{\text{av(vert)}}$ and $\rho_{\text{av(horz)}}$ numerically, to ensure that they are not equal.

Additional constraints must be introduced in order for numerical evaluation of the Holevo capacity to be tractable. The minimum eigenvalue of $\Phi(|\psi\rangle\langle\psi|)$ must be bounded above 0, in order to be able to accurately calculate the matrix logarithm. This constrains the maximum purity of an output state to be bounded away from 1. The highest-purity output state is $\Phi(|0\rangle\langle 0|)$, whose minimum eigenvalue is $1/2(1 - \lambda_z - t_z)$, so we constrain $\lambda_z + t_z = k < 1$. A high value for $\lambda_z + t_z$ is desirable for entanglement preservation and high capacity, however, so we will search over the space for which $k \gtrsim 0.95$. Also, the Holevo quantity χ has zero gradient along the equator of the output ellipsoid, since any state on that equator has the same relative entropy distance from the average than the optimal states on the $z - x$ plane. This will slow a precise numerical calculation of the Holevo capacity if $\lambda_x = \lambda_y$. To ensure that a non-zero gradient for the Holevo quantity exists, we set $\lambda_y < \lambda_x$. Its value is constrained by the CPTP inequalities, which we will use below to constrain it. This leaves two free parameters, k and λ_z. We will search for a pair k, λ_z such that deformations of the resulting channel also possess three-state optimal input sets. Using the entropy expression in Equation 6.14, we can define the degree to which a given channel exhibits the three-state property:

$$\Delta S = S(\Phi(\rho_{\text{av}(3)})) - \frac{1}{2}(S(\Phi(\rho_{\text{av(vert)}})) + S(\Phi(\rho_{\text{av(horz)}})) \tag{6.16}$$

where S is the von Neumann entropy introduced earlier. In order to select a single channel for further study, we will use an algorithm derived by Sam Bader (unpublished):

- Select a value of k.
- Maximize ΔS over λ_z, subject to the constraints that:

- $t_z = k - \lambda_z$
- λ_x is selected such that $C_{\text{horz}}(\lambda_x, \lambda_z, t_z) = C_{\text{vert}}(\lambda_z, t_z)$

Given $\{ \lambda_x, \lambda_z, t_z \}$, select λ_y to saturate the CPTP inequality:

$$(\lambda_x - \lambda_y)^2 = (1 - \lambda_z)^2 - t_z^2 \tag{6.17}$$

This will result in an optimal channel for each value of k. In order to ensure that the minimum eigenvalue of an output density matrix is on the order 10^{-13} (so that it is representable with 3 digits of precision using double-precision floating-point numbers), we select $k = 0.95$, the minimum eigenvalue then being equal to $(1-k)^n 9.8\times \sim 10^{-14}$ for 10 copies of the channel in question. Setting $k = 0.95$ results in a channel with the following parameters:

$$\begin{aligned} \lambda_x &= 0.5364033760658395, \quad \lambda_y = 0.3114370291232779 \\ \lambda_z &= 0.4689014274331901, \quad t_z = 0.4810985725668099 \end{aligned} \tag{6.18}$$

This channel has three optimal inputs:

$$\begin{aligned} &|\tilde{\psi}_1\rangle\langle\tilde{\psi}_1| = |0\rangle\langle 0| = [0, 0, 1], \quad |\tilde{\psi}_2\rangle\langle\tilde{\psi}_2| = [\sin(\theta), 0, \cos(\theta)] \\ &|\tilde{\psi}_3\rangle\langle\tilde{\psi}_3| = [-\sin(\theta), 0, \cos(\theta)], \quad \theta = 1.930429566653508 \end{aligned} \tag{6.19}$$

The optimal ensemble consists of $|\tilde{\psi}_1\rangle$ with probability $p_0 = 0.3622220716889054$, and the states $|\tilde{\psi}_2\rangle$ and $|\tilde{\psi}_3\rangle$ with equal probability $1/2(1 - p_0)$.

Given a preferred channel, and a method of calculating the Holevo capacity, the task which remains is the determination of super-additivity from the result of numerical optimization. In the following section, we go over a few important properties of these numerical results, resulting in methods of observing super-additivity.

6.4 Criteria for Super-Additivity

After convergence is obtained, we are left with a state $|\psi\rangle$ and a locally-optimal relative von Neumann entropy, the value of the pseudo-capacity. In order to determine whether the Holevo capacity of the channel being studied is super-additive, we can examine either the relative entropy obtained, or the pure state corresponding to this optimal relative entropy. As previously discussed, If the pseudo-capacity $\tilde{C}_H(\Phi)$ is greater than $nC_H(\Phi)$,

the Holevo capacity of Φ is super-additive since either $C_{\text{Holv}}(\Phi^{\otimes n}) > nC_{\text{Holv}}(\Phi)$, or the optimal input ensemble contains at least one entangled state. Given that the state $|\psi\rangle$ obtained by performing this optimization corresponds to one of the optimal inputs to $\Phi^{\otimes n}$, we consider states which are not tensor products of the optimal one-qubit states to warrant further study. This is also true in the case that such an entangled state does not achieve the Holevo capacity, since it comprises a local optimum whose entanglement structure we may analyse to gain information for further trials. States from the optimal one-qubit ensemble have $\rho_x = 0, \pm\sin\theta$, so tensor products of these states will have $\text{Tr}(\rho X^{\otimes n})$ equal to either zero, or $\pm\sin^n\theta$. Note that any state of the form $|\psi\rangle = |0\rangle \otimes |\psi_{n-1}\rangle$ will have $\langle\psi|X^{\otimes n}|\psi\rangle = 0$. If the state on the remaining $n-1$ qubits is entangled, it is likely that the study of $n-1$ copies of the same channel would have revealed interesting behaviour on a previous iteration. Therefore, any deviation from this value of $\text{tr}(\rho X^{\otimes n})$ indicates non-standard behaviour, and can be considered indirect evidence of super-additivity. For completeness, we also analyse the entanglement entropy of the optimal input states.

6.5 Numerical Results

In this section, we present tables of numerical optimization results for the Shor optimization defined in the previous section. Before presenting these results, however, we define the 'seed' states used to initialize the optimization routine. Since we wish to study the role entanglement plays in transmitting information through the channel $\Phi^{\otimes n}$, we select entangled seed states, from three families; the GHZ states, the cluster states, and a family of states constructed from the three single-qubit optimal inputs to the channel Φ given in Equation 6.19. For reference, we introduce these states below.

GHZ States The GHZ state [38] can be easily described both in ket notation, and by specifying an n-qubit stabilizer for the n-qubit state:

$$|\psi_{\text{GHZ}}\rangle = \frac{1}{\sqrt{2}}(|0\rangle^{\otimes n} + |1\rangle^{\otimes n}) \tag{6.20}$$

$$S_{\text{GHZ}} = \left\langle X^{\otimes n}, Z_j Z_{j+1} \forall j \in 1 \ldots n-1 \right\rangle \tag{6.21}$$

In order to extend this single state to a basis of 2^n states on n qubits, we multiply the n stabilizers by $\pm\hat{\mathbb{1}}$. The resulting family of states can be described as $1/\sqrt{2}(|b\rangle \pm X^{\otimes n}|b\rangle)$, where b is an n-bit bitstring.

Cluster States The cluster state [66] is another entangled stabilizer state. In contrast to the GHZ state, which is generated by a *non-local* stabilizer (since any generating set must contain some product of $X^{\otimes n}$ with the 'ZZ' stabilizers, which acts on all qubits in the register simultaneously), there exists a generating set for the cluster state which consists of *local* stabilizers:

$$S_{\text{Cluster}} = \langle X_1 Z_2, \{ Z_{k-1} X_k Z_{k+1} \forall k \in 2 \dots n-1 \}, Z_{n-1} X_n \rangle \tag{6.22}$$

To extend this state to a basis for an n-qubit space, we apply the same generalization as before, multiplying elements of the generator by $\pm\hat{\mathbb{1}}$. For this family of states, we use the fact that the projector onto the $+1$-eigenspace of a stabilizer subspace is $1/2^r \prod_{j=1}^{r} (\hat{\mathbb{1}} + S_j)$, where the set $\{ S_j \}$ is a set of generators for the stabilizer in question.

Compressed Qutrit States Both GHZ and cluster states will function as entangled seed states for Shor optimization. However, we also wish to use a set of entangled seed states which take advantage of the known optimal inputs of Φ. Since there are three optimal inputs, we will seed the optimization with a family of 'compressed' qutrit GHZ states. The qutrit GHZ state from which we will construct this family is

$$|\psi_{\text{GHZ-3}}\rangle = \frac{1}{\sqrt{3}} \left(|0\rangle^{\otimes n} + |1\rangle^{\otimes n} + |2\rangle^{\otimes n} \right). \tag{6.23}$$

We extend this state to a larger set, allowing an arbitrary computational qutrit basis state to replace $|0\rangle^{\otimes n}$:

$$|\psi_{\text{GHZ-3}}\rangle \mapsto \frac{1}{\sqrt{3}} \left(|t\rangle^{\otimes n} + (X|t\rangle)^{\otimes n} + (X^2|t\rangle)^{\otimes n} \right), \tag{6.24}$$

where X, here is an operator on qutrits satisfying $X|j\rangle = |j+1 \mod 3\rangle$, and t is an n-trit string. This set of states can then be compressed by applying a map from the qutrit space to the qubit space:

$$|0\rangle_{\text{Qutrit}} \mapsto |\tilde{\psi}_1\rangle, \ |1\rangle_{\text{Qutrit}} \mapsto |\tilde{\psi}_2\rangle, \ |2\rangle_{\text{Qutrit}} \mapsto |\tilde{\psi}_3\rangle. \tag{6.25}$$

6.5.1 Tables of Results

For 2-5 copies of the channel Φ, we input the three families of seed states described above into the Shor optimization algorithm, and analyse the resulting optimal states for their values of $\text{tr}(\rho X^{\otimes n})$, the pseudo-capacities with which they are associated (as defined by Equation 6.4), and their entropies of entanglement [60]. For 6-10 copies of the channel, we select 36 seed states from each of these families.

Maximum Regularized Pseudo-capacity				
	Number of copies			
Input	2	3	4	5
GHZ	0.2443433022720285	0.2453101048673469	0.2457935061650056	0.2455034653864105
Cluster	0.2472437100579757	0.2472437100579800	0.2472437100579816	0.2472437100579822
Compressed	0.2472437100579799	0.2472437100579808	0.2472437100579827	0.2472437100579826

Maximum Regularized Pseudo-capacity				
Number of copies				
6	7	8	9	10
0.2457935061650058	0.2455863341802949	0.2457935061650066	0.2456323723991195	0.2472437100579791
0.2472437100579824	0.2472437100579823	0.2472437100579826	0.2472437100579827	0.2472437100579831
0.2472437100579823	0.2472437100579827	0.2472437100579825	0.2472437100579855	0.2472437100579841

Table 6.1: Pseudo-capacities for Φ. Any value greater than 0.24724371005799 would indicate super-additivity.

Maximum Deviation from $\mathrm{tr}(\rho X^{\otimes n}) = 0, \sin(\theta)^n$			
	Number of copies		
Input	2	3	4
GHZ	$1.238554005633580\times10^{-1}$	$1.110167728450143\times10^{-11}$	$2.323706408780062\times10^{-1}$
Cluster	$8.917387742668481\times10^{-7}$	$4.211726751801592\times10^{-7}$	$1.351877285893366\times10^{-6}$
Compressed	$3.420422567397626\times10^{-13}$	$4.779179762649549\times10^{-16}$	$9.670272043649563\times10^{-16}$

Maximum Deviation from $\mathrm{tr}(\rho X^{\otimes n}) = 0, \sin(\theta)^n$		
Number of copies		
5	6	7
$7.088591641725436\times10^{-12}$	$9.773654740389698\times10^{-12}$	$2.013686297141709\times10^{-13}$
$1.734800370134515\times10^{-6}$	$1.646053666148894\times10^{-6}$	$2.089438061458893\times10^{-6}$
$3.108145995311436\times10^{-16}$	$6.085761903565938\times10^{-16}$	$5.159430292439613\times10^{-16}$

Maximum Deviation from $\mathrm{tr}(\rho X^{\otimes n}) = 0, \sin(\theta)^n$		
Number of copies		
8	9	10
$2.851575367240856\times10^{-13}$	$3.951419311798442\times10^{-14}$	$1.119035159313775\times10^{-6}$
$2.662157798538622\times10^{-6}$	$2.350176267373882\times10^{-6}$	$3.349613489800340\times10^{-6}$
$1.315828520587203\times10^{-6}$	$1.329391249749179\times10^{-6}$	$2.059192139691746\times10^{-6}$

Table 6.2: Deviations from values of $\mathrm{tr}(\rho X^{\otimes n})$ expected for tensor products of single-qubit optimal input states. Non-zero values constitute evidence of an interesting phenomenon, though they do not directly indicate super-additivity.

Maximum Entanglement Entropy									
	Number of copies								
Input	2	3	4	5	6	7	8	9	10
GHZ	1	$< 10^{-13}$	1	$< 10^{-13}$	$< 10^{-13}$	$< 10^{-13}$	$< 10^{-13}$	$< 10^{-13}$	$< 10^{-13}$
Cluster	$< 10^{-13}$	$< 10^{-13}$	$< 10^{-13}$	$< 10^{-13}$	$< 10^{-13}$	$< 10^{-13}$	$< 10^{-13}$	$< 10^{-13}$	$< 10^{-13}$
Compressed	$< 10^{-13}$	$< 10^{-13}$	$< 10^{-13}$	$< 10^{-13}$	$< 10^{-13}$	$< 10^{-13}$	$< 10^{-13}$	$< 10^{-13}$	$< 10^{-13}$

Table 6.3: Bipartite entanglement values for optimal inputs. Non-zero values indicate an entangled local optima, though they do not constitute direct evidence of super-additivity.

6.5.2 Discussion

There are four key observations that we determine from the data above:

1. None of the seed states result in an optimum which shows super-additivity of the Holevo capacity.
2. GHZ seed states result in entangled optima for $n = 2, 4$, but not for any other values of n.
3. GHZ seed states result in deviations in the value of $\mathrm{tr}(\rho X^{\otimes n})$ in the local optima for $n = 2, 4$, but not for any other values of n.
4. Cluster seed states result in increasing values of the deviation in the value of $\mathrm{tr}(\rho X^{\otimes n})$ in the local optima.

In the following section, we draw conclusions from this data, and discuss additional research to be conducted.

6.6 Conclusions and Future Work

No direct numerical observation of the super-additivity of the Holevo capacity has been made. However, the research explained in this chapter comprises a useful first step in the search for concrete numerical examples of channels (or sets of channels) whose Holevo capacities are super-additive. Future research will likely focus on increasing the number of channel copies n. In order to calculate Holevo capacities for higher n, this research will need to use a more efficient representation of the interim state $\Phi^{\otimes n}(|\psi_k\rangle\langle\psi_k|)$, and the matrix $\log(\Phi^{\otimes n}(\rho_a v^{\otimes n}))$, as memory is the limiting factor in determining the maximum n. It is possible that, due to the tensor-product structure of $\log(\Phi^{\otimes n}(\rho_a v^{\otimes n}))$, it can be efficiently subtracted from $\log(\Phi^{\otimes n}(|\psi_k\rangle\langle\psi_k|))$ without storing it. Taking this into account, and only storing the upper-triangular portion of $\Phi^{\otimes n}(|\psi_k\rangle\langle\psi_k|)$ (which is Hermitean) would reduce the amount of memory required by a factor of 4, allowing an calculation at $n = 12$. It will also be necessary to transfer the Holevo capacity calculation algorithm to a large computational cluster, where the amount of memory is greater by a factor of $4-10$, possibly allowing for calculations of the capacity at $n = 13 - 14$.

Chapter 7

Quantum Transcoding

Another important problem in quantum computing is the design of protocols to perform universal computation on data which has been encoded into a quantum error-correcting code, so-called *fault-tolerant quantum computation*. The stabilizer formalism discussed in Chapter 3 can be used to efficiently design error-correcting codes, but the implementation of a universal set of logical operations has historically required additional resources, such as ancillary systems prepared in non-stabilizer states [12]. In this chapter, we summarize the history of development of fault-tolerant computing, then introduce *quantum transcoding*, the practice of transferring data between quantum error-correcting codes in order to perform individual logical operations within codes in which they are fault-tolerant. We conclude with a preliminary result which shows the utility of a simple transcoding operation, not for implementing a universal gate set, but for increasing the memory threshold of a QECC.

7.1 History of Fault-Tolerant Computing

The development of individual quantum codes to correct local Pauli errors [77, 81, 55], beginning in 1994, gave way to a single framework which unified the known quantum codes, the stabilizer formalism ([36], also see Chapter 3). This unification provided a general means of defining the logical gates which act on an encoded register. Specifically, since the stabilizer of a given code is a commutative subgroup of the Pauli group, the operations which map the Pauli group onto itself are a group which contains logical operations. This set of operations forms a finite group (the Clifford group), which cannot be used for universal computation.

As a result, development of universal fault-tolerant computing using stabilizer codes shifted its focus toward methods of implementing the Clifford group in a fault-tolerant fashion, then extending it to a universal set of logical gates. Throughout this phase of development, it was often possible to find transversal implementations of the elements of the Clifford group, with some codes having transversal implementations of the entire Clifford group. Later, individual stabilizer codes were found with transversal implementations of specific non-Clifford operators [84], but at no point was there a single code which possessed a transversal, universal set of logic gates.

This gave rise to a conjecture that no stabilizer code could admit such a set, which was proven for stabilizer codes on qubits in 2007 by Zeng, et al. [90]. This proof employed a contradiction, showing that a transversal universal gate set could map a logical Pauli to a Pauli of weight less than the distance of the relevant stabilizer code. This result was later extended by Chen et al [15] to stabilizer codes on quantum systems of arbitrary dimension, and by Eastin and Knill [28] to any quantum code, independent of its representability in the stabilizer formalism, which can detect local errors.

There remained, and exist today, avenues to circumvent these results. The first and most widely studied of these is gate teleportation [37, 91], which, in conjunction with the preparation of Clifford eigenstates, can be used to implement non-Clifford logic gates. These Clifford eigenstates (also called 'magic states') can be prepared iteratively, first preparing a large number of magic states which are subject to noise, then *distilling* from this large set of states a small number of high-fidelity states. The process of magic state distillation has been the subject of much research [12, 11, 62]. The magic state distillation schemes put forth thus far have been extremely expensive [31], prompting research into alternate means of circumventing the no-go result.

Another means to implement fault-tolerant computation hinges on a detail of the proofs given in [90, 15, 28], that all two-qubit transversal gates are transversal with respect to the same partition of a code register. That is, if for every qubit j in a code block there exists a qubit j' in a second code block and a transversal operation couples qubits j and j', the relationship between the indices j and j' does not depend on the gate being implemented. There exist codes which violate this criterion, such as the Bacon-Shor codes [6, 4], which use re-labellings of the qubits in order to implement the logical Hadamard, although this does not lead to universality.

A promising result in fault-tolerant computing without gate teleportation was recently put forth by Paetznick and Reichardt [62]. They show that a known $[\![15, 7, 3]\!]$ quantum error-correcting code allows a transversal Hadamard gate, subject to the constraint that 6 of the 7 logical qubits are unused, since the transversal implementation of the Hadamard

gate only corresponds to the logical Hadamard gate on the remaining qubit. This code also allows a transversal implementation of the doubly-controlled Z gate, which is not a Clifford gate, subject to the constraint that the same set of 6 logical qubits is kept in the state $|0_L\rangle^{\otimes 6}$. This scheme, while it proves that fault-tolerant universal computation can be obtained using only transversal operations and established error-correcting techniques, does not represent a feasible protocol in terms of its error threshold which, while not rigorously bounded, is estimated to be below 10^{-4}.

These fault-tolerance schemes vary greatly in their error thresholds and levels of overhead. It is not clear, a priori, which approach is preferable for creating fault-tolerant universal gate sets. There remains a largely-unexplored avenue for circumventing the no-go results [90, 15, 28], if information in a quantum error-correcting code can be fault-tolerantly transferred to another code, each possessing transversal gate sets whose union is universal. This is quantum transcoding, which we outline in the next section.

7.2 Quantum Transcoding

In order to define an operation which transfers information between codes, we first examine the process of encoding one qubit into an $[\![n, 1, d]\!]$ code:

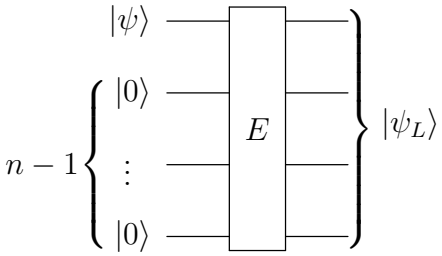


Figure 7.1: A generic encoder for a stabilizer code, taking $|\psi\rangle \otimes |0\rangle^{\otimes n-1}$ to $|\psi_L\rangle$.

The stabilizer group which defines the state before it is encoded is generated by $\{ Z_j \mid 2 \leq j \leq n \}$, operators which constrain the state of the $n-1$ ancilla qubits to be $|0\rangle^{\otimes n-1}$, while leaving the first qubit in an arbitrary state $|\psi\rangle$. After the encoding operation is complete, the stabilizer group is that for the code in question. Likewise, the logical Paulis prior to encoding are simply X_1 and Z_1, encoding maps them to X_L, Z_L. We can specify the action of the

encoder in Clifford notation, describing its action on the Pauli group generators:

$$\begin{matrix} IZ\cdots I \\ \vdots \\ II\cdots Z \end{matrix} \overset{E}{\longmapsto} \begin{matrix} S_1 \\ \vdots \\ S_{n-1} \end{matrix}, \quad ZI\cdots I \overset{E}{\longmapsto} Z_L, \quad XI\cdots I \overset{E}{\longmapsto} X_L \tag{7.1}$$

We note that the action of a decoding Clifford can be specified by reversing the maps given in Equation 7.1.

There is no unique encoder for a given stabilizer code. The action of the encoder is unspecified on the generators $\{ X_j \mid 2 \leq j \leq n \}$. We are free to select an arbitrary set of Paulis to assume the roles of $\{ \bar{X}_j \mid 2 \leq j \leq n \}$, as long as they preserve the commutation relations between $\{ X_j \mid 2 \leq j \leq n \}$ and the other generators. Specifically:

- Each Pauli $\bar{X}_j$ must commute with $\bar{X}_1 = X_L$, the logical X for the code, and all other elements of $\{ X_j \mid 2 \leq j \leq n \}$.
- Each Pauli $\bar{X}_j$ must anti-commute with the corresponding $\bar{Z}_j$.
- Each Pauli $\bar{X}_j$ must be independent from the rest of the set $\{ \bar{X}_j \mid 1 \leq j \leq n \}$, no product of the set $\{ \bar{X}_k, k \neq j \}$ being equal to $\bar{X}_j$.

The number of Pauli sets which fulfil these criteria is equal to $\prod_{j=2}^{n} N_j$ where N_j is the number of satisfying assignments for the j^{th} Pauli in the set. The total size of the Pauli group on n qubits is 2^{2n}. Anti-commutation with $\bar{Z}_j$ restricts the size of the set from which $\bar{X}_j$ can be drawn from to size 2^{2n-1}. Commutation with $\{ \bar{Z}_k \mid k \neq j \}$ further reduces this size to 2^n. Thinking of the process of selecting these Paulis as being serial, the j^{th} Pauli must commute with $j-1$ Paulis. The constraint that $\bar{X}_j$ not be generated by $\{ \bar{X}_k \mid k < j \}$ is automatically fulfilled, since no Pauli generated by that set can anti-commute with $\bar{Z}_j$, and $\bar{X}_j$ is constrained to do so. Therefore, $N_j = 2^{n-j+1}$, and the number of Pauli sets which can be selected to constrain the degrees of freedom for an encoder is

$$\prod_{j=2}^{n} 2^{n-j+1} = 2^{\frac{n(n-1)}{2}}. \tag{7.2}$$

This function grows extremely rapidly, attaining a value in excess of 35 trillion for 10 qubits. This restricts any search over encoding operations to small numbers of qubits, $\sim 5-7$.

Transcoding operations between a pair of codes (called C and C' here) can be defined analogously to encoders. There are three cases which we must consider; $n_1 = n_2$, $n_1 < n_2$ and $n_1 > n_2$:

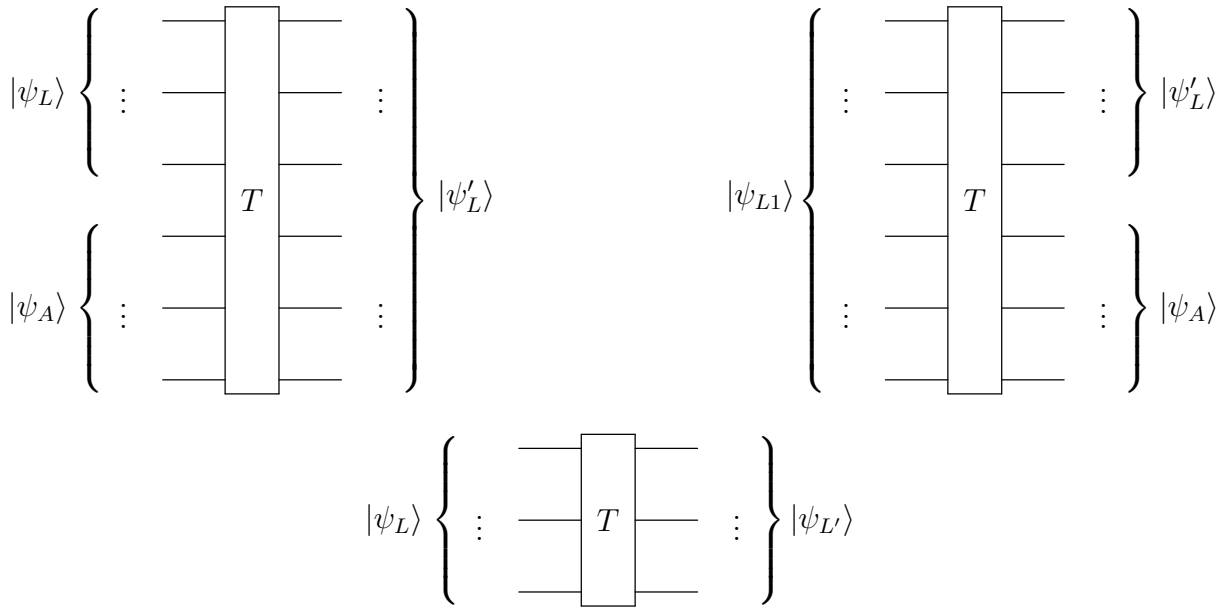


Figure 7.2: A generic transcoder for a stabilizer code, taking $|\psi_L\rangle$ to $|\psi'_L\rangle$, involving an ancillary stabilizer state in the case that $n < n'$, yielding a measurement result in the case $n > n'$, and having no effect outside the encoded register when $n = n'$.

The action of a transcoder on a generating set of the Pauli group can also be analysed. The initial generating set consists of a stabilizer generator, pair of logical Paulis and $n-1$ $\bar{X}$ operators (the unconstrained degrees of freedom in the encoder), and the final generating set consists of the stabilizer, logical operators, and free operators of C'. We constrain both codes to have the same number of logical qubits ($k = k' = 1$ in the examples considered here), and constrain $X_L \overset{T}{\longmapsto} X'_L$, $Z_L \overset{T}{\longmapsto} Z'_L$. In the case where $n = n'$, the Clifford relation is easy to define:

$$\begin{array}{ccc} S_1 & & S'_1 \\ \vdots & & \vdots \\ S_{n-1} & & S'_{n-1} \\ X_L & \overset{T}{\longmapsto} & X'_L \\ \bar{X}_1 & & \bar{X}'_1 \\ \vdots & & \vdots \\ \bar{X}_{n-1} & & \bar{X}'_{n-1} \\ Z_L & & Z'_L \end{array} \tag{7.3}$$

Each such transcoder construction must satisfy the following constraints:

- A generating set S for the stabilizer of the initial code must be mapped to another generating set S' for the output code.
- $X_L \overset{T}{\longmapsto} X'_L$, $Z_L \overset{T}{\longmapsto} Z'_L$, as discussed earlier.
- A generating set for the free operators $\{ \bar{X} \}$ must be mapped to another generating set $\{ \bar{X}' \}$.

The number of operators which satisfy all of these constraints is the product of the number of operators satisfying the constraints individually, since they are independent.

To count the generating sets of the stabilizer for an $[\![n, 1, d]\!]$ code, we construct an arbitrary generating set from the 2^{n-1} elements of the stabilizer group, by selecting independent Paulis in series. To select the j^{th} Pauli, we must not select one generated by the $j-1$ Paulis selected prior to the j^{th}, including the identity. Thus, there are $2^{n-1} - 2^{j-1}$ Paulis which can be selected at the j^{th} step, and $\prod_{j=1}^{n-1} 2^{n-1} - 2^{j-1}$ generating sets for the stabilizer code C. Note that, if S is one of the sets selected according to the scheme presented here, all permutations of S are also generated by this scheme. Since the definition of a Clifford as a map from Paulis to Paulis does not depend on the order in which the inputs and outputs are enumerated, we divide by the number of permutations of S, counting permutations of S' relative to S, to obtain $\left(\prod_{j=1}^{n-1} 2^{n-1} - 2^{j-1}\right)^2 / (n-1)!$ as the number of distinct stabilizer-to-stabilizer maps.

Each of the logical operators, in turn, can be replaced by a member of the set consisting of products of the logical operator in question with an element of the stabilizer group (known in group theory as the *coset* of the stabilizer). Since there are two logical Paulis and 2^{n-1} elements of the stabilizer group, there are 2^{2n-2} permissible logical Pauli sets for each of the codes C and C', with 2^{4n-4} maps between permissible sets.

As discussed earlier, there are $2^{\frac{n(n-1)}{2}}$ satisfying assignments for the set $\{ \bar{X} \}$ and $\{ \bar{X}' \}$. Identical permutations of two of the satisfying assignments result in an identical map from Paulis to Paulis, so we divide by $(n-1)!$ again. The total number of transcoding Cliffords between two codes on n qubits is then:

$$2^{n(n-1)} 2^{4n-4} \frac{\left(\prod_{j=1}^{n-1} 2^{n-1} - 2^{j-1}\right)^2}{((n-1)!)^2}$$

For five qubits, this number is in excess of 48 quadrillion, prohibiting a direct search over the entire set of transcoders for those that are fault-tolerant. The criterion we use for fault-tolerance assumes that both the input and output codes have distance 3, so that weight-one

errors must be mapped by T to weight-one errors. Since there exists no two-qubit gate for which this is the case, such a transcoding operation would have to be transversal. However, since the overall goal of transcoding is to facilitate universal computation, either the transcoder or one of the logical gates used by the input or output code must be non-transversal. Since we wish to use transversal logical gates, we must generalize to the case $n \neq n'$.

We elect to study the case $n > n'$, since output faults of weight greater than one on the n-qubit register after transcoding are still correctable, if their support on the n' qubits on which data still resides is one or less. Transcoders between C and C' must satisfy a set of constraints when $n > n'$ which is similar to those satisfied for $n = n'$. For instance, it is still the case that $k = k' = 1$, so there exist $2^{2n-2} \times 2^{2n'-2}$ permissible maps from logical Paulis to logical Paulis. There still exist the same number of free operators for both the input and output to T, so the number of permissible maps from free operators to free operators is $2^{n(n-1)}$.

The constraint that T maps a generating set of the input stabilizer to a generating set of the output requires additional work to analyse, since the number of generators for C' is less than that for C. Also note that the output state must be separable, so the output stabilizers must either be supported on the first n' of the n-qubit register, or the last $n-n'$. The map acts on the input stabilizers as follows:

$$\begin{matrix} S_1 \\ \vdots \\ S_{n-1} \end{matrix} \overset{T}{\longmapsto} \begin{matrix} S'_1 \otimes \hat{\mathbb{1}}^{\otimes n-n'} \\ \vdots \\ S'_{n'-1} \otimes \hat{\mathbb{1}}^{\otimes n-n'} \\ \hat{\mathbb{1}}^{\otimes n'} \otimes P'_1 \\ \vdots \\ \hat{\mathbb{1}}^{\otimes n'} \otimes P'_{n-n'} \end{matrix} \tag{7.4}$$

The number of satisfying assignments for the output of the map changes to the product of the number of generating sets for S' and the number of sets of $n - n'$ commuting, independent Paulis that can be found on $n - n'$ qubits. The number of generating sets for S' is, as calculated earlier, $\prod_{j=1}^{n'-1} 2^{n'-1} - 2^{j-1}$. The number of satisfying assignments for the operators P can be derived using a similar iterative calculation. When selecting the j^{th} Pauli, one can select any of the 2^{2n} members of the Pauli group, subject to $j-1$ commutation constraints, selecting none of the 2^{j-1} Paulis which are in the generated group of the $j-1$ Paulis selected thus far. Therefore, the number of satisfying assignments to this set is $\prod_{j=1}^{n-n'} 2^{2(n-n')-j} - 2^{j-1}$. All permutations of the individual sets S' and P are included, but the calculation thus far assumes that the elements of S' are listed, followed

by the elements of P. This need not be the case, elements of P can be interspersed with elements of S' in any order. The number of ways in which this can be accomplished is the *multicombination*, $\left(\binom{n'}{n-n'}\right) = \binom{n-1}{n-n'}$.

Performing the same product as for the case $n = n'$, then, the number of transcoding Cliffords for the case $n > n'$ is:

$$2^{n(n-1)}2^{2n-2}2^{2n'-2}\frac{\left(\prod_{j=1}^{n-1} 2^{n-1} - 2^{j-1}\right)\left(\prod_{j=1}^{n'-1} 2^{n'-1} - 2^{j-1}\right)\left(\prod_{j=1}^{n-n'} 2^{2(n-n')-j} - 2^{j-1}\right)}{(n-n')!(n'-1)!}$$

For $n = 5$, $n' = 3$, the number of possible transcoders is approximately 1.8×10^{15}. For any transcoding problem of interest, the size of this set makes a complete search prohibitively expensive, especially considering that the number of potential transcoding Cliffords is the product of its size with the size of the set of valid outputs. Since the set of transcoding Cliffords cannot be directly searched, there are two options available. We can either place additional restrictions on the input and output sets of a given transcoding operation, or we can restrict the form of the transcoder in another fashion. In the following subsections, we explore each of these options in turn, ending with a preliminary result in enhancing the memory thresholds of CSS codes [82].

7.3 Restricted Search

The number of Cliffords which perform a given transcoding operation is extremely large. In order to ensure that a transcoding operation can be fault-tolerantly implemented, we examine its circuit decomposition to determine if faults in the circuit propagate adversely. However, if there exist a large fraction of transcoding operations which admit fault-tolerant circuit decompositions, we may be able to locate one of them by examining only a small number of transcoding Cliffords. For this reason, we perform a restricted search, fixing the input/output stabilizer generators and logical operators, searching over the free operators $\{\bar{X}\}$ and $\{\bar{X}'\}$, and the values of the output ancilla set P. This reduces the number of Cliffords to

$$2^{n(n-1)} \prod_{j=1}^{n-n'} 2^{2(n-n')-j} - 2^{j-1}$$

We focus on three specific transcoding tasks; taking the 'perfect' 5-qubit code to the three-bit repetition code versus bit-flip, taking the perfect 5-qubit code to the three-bit repetition

code versus phase-flip, and taking the Steane 7-qubit code to the perfect code. These tasks have been considered in the past, with measurement-based gadgets having been derived [85, 41]. The number of Cliffords in the restricted set is approximately 14 million for $n = 5$, $n' = 3$, 28 trillion for $n = 7$, $n' = 5$. While only a small fraction of the transcoders taking the Steane code to the perfect code can be searched, a complete search over the restricted set can be executed for the transcoders taking the perfect code to either of the three-bit repetition codes.

In order to analyse the output Cliffords, we find their circuit decompositions using an algorithm defined by Aaronson and Gottesman [2], implemented in QuaEC, a Python library for quantum error correction (for an introduction to QuaEC, see Appendix C or [19]). To determine which, if any, of the found transcoding Cliffords are fault-tolerant, we count the number of two-qubit gates in the circuit decomposition. We show, below, the circuits output by the search which have the minimal number of two-qubit gates:

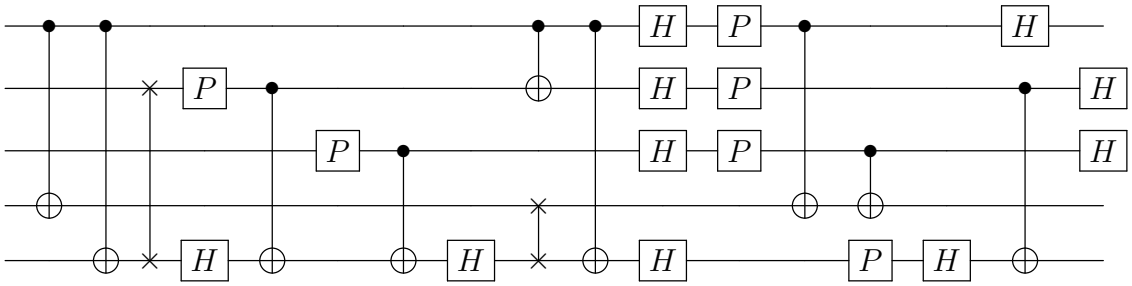


Figure 7.3: A transcoding circuit taking the 5-qubit perfect code to the 3-qubit repetition code correcting bit-flip, containing the minimal number of error-propagating two-qubit gates (nine).

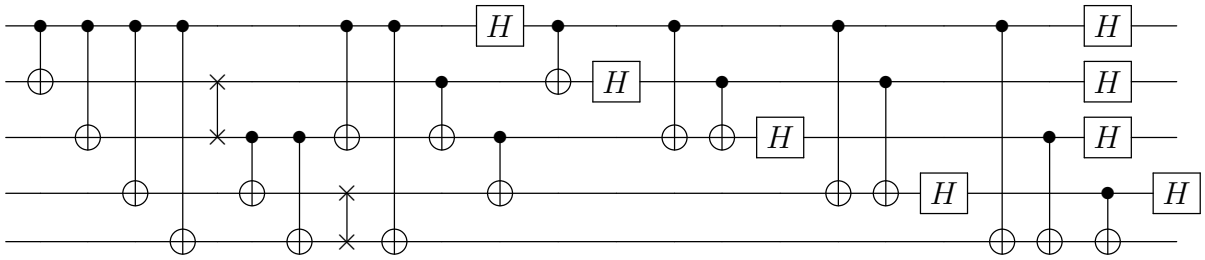


Figure 7.4: A transcoding circuit taking the 5-qubit perfect code to the 3-qubit repetition code correcting phase-flip, containing the minimal number of error-propagating two-qubit gates (eighteen).

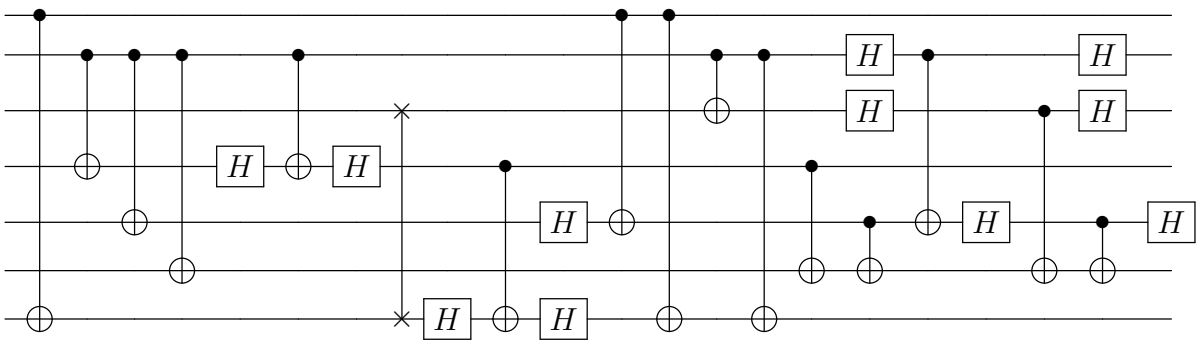


Figure 7.5: A transcoding circuit taking the 7-qubit Steane code to the 5-qubit perfect code, containing the minimal number of error-propagating two-qubit gates (fifteen).

It is evident that restricting the search in this manner does not lead to fault-tolerant circuits, given that errors can propagate adversely across any of the quantum circuits given above. We advance, in the next section, to restricting the form of the output circuit directly, deriving a transcoding circuit which, while not facilitating universal quantum computing, assists in storing quantum information in CSS codes.

7.4 Transversal Transcoding & Memory Threshold Enhancement

It is prohibitively expensive to construct the set of transcoding Cliffords and search through that set for Cliffords which can be decomposed into fault-tolerant circuits. In this section, we analyse the opposite approach; searching sets of fault-tolerant Cliffords for those which execute transcoding operations.

One such set can be constructed analytically. We note that the stabilizers of the CSS codes share an important trait with that for the Steane 7-qubit code:

$$S_{\text{Steane}} = \langle G \rangle \, , \, G = \begin{matrix} XXXXIII \\ XXIIXXI \\ XIXIXIX \\ ZZZZIII \\ ZZIIZZI \\ ZIZIZIZ \end{matrix} \, , \tag{7.5}$$

namely, their generating sets can be partitioned into two sets, one containing stabilizers consisting of tensor products of the Paulis X and I, the other consisting of tensor products

of Z and I. The correctable errors for distance-3 CSS codes are then X, Y and Z-errors on any individual qubit, as well as weight-two errors consisting of one X and one Z on distinct qubits. This increases the ability of a CSS code to correct X and Z errors, relative to the ability to correct Y errors. This ability can be easily analysed using the effective Pauli-basis error map introduced by Rahn, Doherty and Mabuchi [65], which treats the error-correction process as a quantum channel acting on the 2-dimensional Hilbert space of the logical qubit. The effective noise parameters in the x, y, and z-directions (these being the diagonal elements of the noise superoperator in the Pauli-basis, see Chapter 2 for an introduction) are given for the Steane code:

$$\begin{aligned}&[x,\, y,\, z] \underset{\text{Steane}}{\longmapsto} [S(x),\, T(x,y,z),\, S(z)]\,,\\ &\text{where } S(x) = \frac{7}{4}x^3 - \frac{3}{4}x^7, \qquad T(x,y,z) = \frac{7}{16}y^3 + \frac{9}{16}y^7 - \frac{21}{16}(x^4+z^4)y^3 + \frac{21}{8}x^2yz^2\end{aligned} \tag{7.6}$$

For isotropic noise ($x = y = z = r$, $r = 1 - \frac{4}{3}p$ for uniform error probability p), we can calculate a threshold value of p, p_{th}, by taking the maximum value of the fixed points of the maps $r \mapsto S(r)$, and $r \mapsto T(r)$. The map $r \mapsto S(r)$ has a fixed point at $r_* \sim 0.8708$, and $r \mapsto T(r)$ has a fixed point at $r_* \sim 0.7639$. Since $r = 1 - \frac{4}{3}p$, the value of p_{th} is 0.0969, being dominated by the x and z terms, both of which are susceptible to Y errors. The corresponding value of the threshold probability for Y-eigenstates is ~ 0.1771, approximately 82% greater. Therefore, it is desirable to 'symmetrize' the performance of the code, rendering its thresholds equal for X, Y and Z noise.

This symmetrization can be accomplished by a transversal Clifford, implemented immediately after error correction, which alters the stabilizer as follows:

$$T = \begin{matrix} X \mapsto Y \\ Y \mapsto Z \end{matrix}\,, \tag{7.7}$$

$$T^{\otimes n}(G_{\text{Steane}}) = \begin{matrix} YYYYIII \\ YYIIYYI \\ YIYIYIY \\ XXXXIII \\ XXIIXXI \\ XIXIXIX \end{matrix}\,,\; T^{\otimes n}(T^{\otimes n}(G_{\text{Steane}})) = \begin{matrix} XXXXIII \\ XXIIXXI \\ XIXIXIX \\ YYYYIII \\ YYIIYYI \\ YIYIYIY \end{matrix} \tag{7.8}$$

If we assume the execution of the T gate here to be perfect (or absorb it into the action of the error-correcting gadget), we can replace the action of the error-correction gadget/-transcoder with perfect decoding and re-encoding into the new code. For 3 iterations

of this scheme, we compare the unencoded channel $[r^3, r^3, r^3]$ to the encoded channel $[S(r)S(r)T(r), S(r)T(r)S(r), T(r)S(r)S(r)]$. The map $r \mapsto S(r)S(r)T(r)$ has the fixed point $r_* = 0.8018$, which implies $p_{th} \sim 0.1487$, a value 53% larger than the threshold without having transcoded. Similar gains in memory threshold can also be had for other codes which exhibit similar susceptibility to one type of Pauli error.

7.5 Conclusions and Future Work

The utility of transcoding as a means of obtaining fault-tolerant gate sets is still in question, due to the immense size of the Clifford group on n qubits. However, there exist a few simple circuits which prove that transcoding has some benefits to quantum error-correction. The next avenue of research to pursue is the expansion of the circuit search set to include circuits possessing a limited number of two-qubit Clifford gates, examining this set for those which perform transcoding operations, since these circuits number far fewer than the number of Clifford operators which perform a transcoding task. In any event, quantum transcoding remains a relatively unexplored area of research with interesting results ahead.

Chapter 8

Conclusion

The development of quantum information processing has reached an exciting new stage. Research continues into physical systems which allow storage and manipulation of quantum information with ever-higher speed and fidelity, and into error-correction schemes which permit the simulation of perfect operations using fewer, noisier physical qubits. The state of the art in error-correction research is quite sophisticated, having advanced from proofs of principle to analysis and improvement of performance, with current research being focused on schemes for universal computation, and families of codes exhibiting common properties.

At this stage of research, marginal gains in performance become more important, and it becomes useful to study ways in which more can be done with less. This is the overall theme of the research in this thesis. We have discussed, in Chapter 5 means of partially correcting additional errors without adding ancilla qubits to a quantum error-correcting scheme. In Chapter 6, we attempt to determine whether more classical information can be transmitted through a composite channel using entangled states. In Chapter 7, we show a means of increasing the memory performance of CSS codes using no additional ancilla qubits, and discuss an extension of the techniques involved to performing universal encoded computation.

In addition, we detail two useful tools for performing calculations related to quantum error correction and communication. In Appendix B we introduce a more efficient way to calculate the effect of a CPTP map on a density matrix, given that the map acts on one qubit. In Appendix C, we introduce a Python library for performing calculations involving Pauli and Clifford operators. We believe that these tools, along with the research detailed in this thesis, will aid the development of quantum computing in the near future.

Appendix A

Semi-Definite Programming for QEC

In Chapter 5, we considered the task of correcting quantum errors when an additional source of noise is present; specifically, the presence of initialization noise on the ancilla qubits. To accomplish this, we rephrased the problem of maximizing channel fidelity for a given error-correcting code as a semi-definite program, following the method of [68]. In this appendix, we provide a brief introduction to semi-definite programs, and a derivation showing that, with certain restrictions, quantum error correcting codes can be designed using semi-definite programs. We conclude by detailing a method for placing additional constraints on the encoding and decoding maps, ensuring that they are unital.

A.1 Semi-definite Programs

A semi-definite program is an optimization problem of the following form:

$$\begin{aligned} &\min_{x} \langle c|x\rangle \\ &\text{such that } A\,|x\rangle = |b\rangle \end{aligned} \tag{A.1}$$

where $|c\rangle$, $|x\rangle$, and $|b\rangle$ are members of a *convex cone* (a subset of a vector space with closure under convex combinations using positive coefficients, see [68] and references), such as the positive semi-definite matrices. Here, the bra-ket notation is meant to denote abstract inner products, and the matrix-vector constraint $A\,|x\rangle = |b\rangle$ denotes the satisfaction of numerous linear constraints by $|x\rangle$.

Semi-definite programs can be solved in an amount of time which scales as n_x^3, where n_x is the number of parameters in $|x\rangle$. There are numerous free software packages which

solve semi-definite programs, such as CVXOPT [59] and SeDuMi [87]. The reduction of the derivation of quantum-error-correcting codes to the solution of semi-definite programs thus provides access to a well-developed set of optimization algorithms.

In the following section, we provide a brief derivation of the semi-definite programs whose solutions represent locally-optimal quantum-error-correcting operations with respect to channel fidelity. A comprehensive treatise on this derivation can be found in [68]; it is included here because we will have to include a non-standard constraint, that the encoding/decoding maps are unital. The reason for this constraint (that error correcting operations are not permitted to increase the purity of an input state) is explained in detail in the final section.

A.2 Optimal Encoders/Decoders using SDP

We consider an error-correction procedure to consist of three steps; encoding E, transmission through a noisy channel T, and decoding D (we encapsulate any error-correction operations which perform syndrome-controlled operations on the decoded qubit in D). For either of the two maps E and D, but not both simultaneously, we can express the channel fidelity F_C as a matrix inner product, exploiting the cyclic property of the trace, and the existence of dual channels (introduced in Chapter 4):

$$\begin{aligned} F_C(D \circ T \circ E) &= \langle\Omega| D \circ T \circ E \otimes \hat{\mathbb{1}} \left[|\Omega\rangle\langle\Omega|\right] |\Omega\rangle \\ &= \mathrm{tr}(\langle\Omega| D \circ T \circ E \otimes \hat{\mathbb{1}} \left[|\Omega\rangle\langle\Omega|\right] |\Omega\rangle) \\ &= \mathrm{tr}(|\Omega\rangle\langle\Omega| \cdot D \circ T \circ E \otimes \hat{\mathbb{1}} \left[|\Omega\rangle\langle\Omega|\right]) \\ &= \mathrm{tr}((D \circ T \otimes \hat{\mathbb{1}})_* \left[|\Omega\rangle\langle\Omega|\right] E \otimes \hat{\mathbb{1}} \left[|\Omega\rangle\langle\Omega|\right]) \end{aligned} \tag{A.2}$$

where $|\Omega\rangle = \frac{1}{\sqrt{d}} \sum_j |j\rangle \otimes |j\rangle$, as defined in Chapter 2, and Λ_* is the dual channel to Λ, this duality being defined in Chapter 4.

This matrix inner product can be expressed as a vector inner product, since:

$$\mathrm{tr}(A^\dagger B) = \sum_j (A^\dagger B)_{jj} = \sum_{jk} A^\dagger_{jk} B_{kj} = \langle \mathrm{col}(A) | \mathrm{col}(B) \rangle \,. \tag{A.3}$$

Therefore, the channel fidelity can be expressed in the semi-definite programming form, an inner product of column-stacked Choi matrices:

$$F_C(D \circ T \circ E) = \langle \mathrm{col}((D \circ T \otimes \hat{\mathbb{1}})_* \left[|\Omega\rangle\langle\Omega|\right]) | \mathrm{col}(E \otimes \hat{\mathbb{1}} \left[|\Omega\rangle\langle\Omega|\right]) \rangle \,. \tag{A.4}$$

A similar relation can be seen to hold for the decoding map:

$$F_C(D \circ T \circ E) = \langle \mathrm{col}((T \circ E \otimes \hat{\mathbb{1}})_* [|\Omega\rangle\langle\Omega|]) | \mathrm{col}(D \otimes \hat{\mathbb{1}} [|\Omega\rangle\langle\Omega|]) \rangle \,. \tag{A.5}$$

Thus, in order to perform joint optimization, we alternately optimize over E and D, until a sufficient threshold of convergence has been met. This does not guarantee global optimality over the set E, D. However, numerous codes have been found using this method which exceed the performance of known codes [30, 56].

In order to place the familiar trace-preserving constraint (expressed in the Kraus formalism),

$$\sum_k A_k^\dagger A_k = \hat{\mathbb{1}},$$

in the semi-definite programming form $A |x\rangle = |b\rangle$, we use the map between the Kraus and Choi representations of a quantum channel:

$$\begin{aligned} \Phi_\Lambda &= \Lambda \otimes \hat{\mathbb{1}} [|\Omega\rangle\langle\Omega|] \\ &= \sum_k A_k \otimes \hat{\mathbb{1}} |\Omega\rangle\langle\Omega| A_k^\dagger \otimes \hat{\mathbb{1}} \\ &= \sum_k |\mathrm{col}(A_k)\rangle\langle\mathrm{col}(A_k)| \,. \end{aligned} \tag{A.6}$$

We also use the identity

$$\mathrm{tr}_1 |\mathrm{col}(A_k)\rangle\langle\mathrm{col}(A_k)| = A_k^\dagger A_k,$$

resulting in

$$\mathrm{tr}_1(\Phi_\Lambda) = \hat{\mathbb{1}} \tag{A.7}$$

as the CPTP constraint on Λ (where, here, Λ will be E or D). The partial trace can be expressed in Kraus notation, and its action can be expressed in the column-stacked notation, resulting in a constraint of the appropriate form:

$$\mathrm{tr}_1(\rho) = \sum_k \langle k| \otimes \hat{\mathbb{1}} \rho |k\rangle \otimes \hat{\mathbb{1}} \tag{A.8}$$

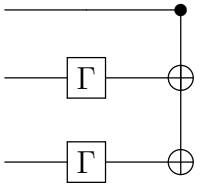
$$|\mathrm{col}(PQR)\rangle = (R^T \otimes P) |\mathrm{col}(Q)\rangle$$

$$\therefore |\mathrm{col}(\mathrm{tr}_1(\Phi_\Lambda))\rangle = \left(\sum_k (\langle k| \otimes \hat{\mathbb{1}}) \otimes (\langle k| \otimes \hat{\mathbb{1}}) \right) |\mathrm{col}(\Phi_\Lambda)\rangle = |\mathrm{col}(\hat{\mathbb{1}})\rangle \,. \tag{A.9}$$

With $A = \left(\sum_k (\langle k| \otimes \hat{\mathbb{1}})^{\otimes 2}\right)$ and $|b\rangle = |\mathrm{col}(\hat{\mathbb{1}})\rangle$, the CPTP constraint is expressed in the standard SDP form. In this way, quantum codes can be derived automatically, and their properties can be studied.

A.3 Unitality

It is important to note that, when selecting E and D, it is typical to restrict E and D to be unitary, since unitary control is to be used for encoding and decoding. However, the unitarity constraint is non-linear, so it cannot be included in the semi-definite program. This implies that, for example, if initialization noise is present (see Chapter 5), the encoder, which is an arbitrary CPTP channel, can take the following form:



where the channel Γ maps all states to $|0\rangle$. Γ can be though of as a generalized amplitude damping channel (see Chapter 2), and is thus CPTP. In order to find the optimal unitary operator, we use an extra semi-definite constraint to derive a bound for unital channels, then attempt to saturate that bound with a unitary encoder. The unital constraint is closely related to the CPTP constraint:

$$\sum_k A_k A_k^\dagger = \hat{\mathbb{1}}, \therefore A = \begin{bmatrix} \left(\sum_k (\langle k| \otimes \hat{\mathbb{1}})^{\otimes 2}\right) \\ \left(\sum_k (\hat{\mathbb{1}} \otimes \langle k|)^{\otimes 2}\right) \end{bmatrix}, |b\rangle = \begin{bmatrix} |\mathrm{col}(\hat{\mathbb{1}})\rangle \\ |\mathrm{col}(\hat{\mathbb{1}})\rangle \end{bmatrix},$$

where the rows of A and $|b\rangle$ are vertically concatenated, since both CPTP and unitality constraints are to be satisfied. The augmentation in Chapter 5 is the result of such a constrained optimization.

A.4 Conclusions

Semi-definite programming is a useful method for studying approximate quantum error correction. It is likely that exact error correction will remain the prevalent avenue of research into quantum error correction, due to the ease with which encoding/decoding operations can be decomposed into circuits whose fault-tolerant properties can be analysed. However, approximate error correction, and optimal codes specifically, will likely come into greater use as low-level error-correction routines which can be adapted to the characteristics of individual implementations of quantum computing.

Appendix B

Numerically-Efficient Application of Quantum Channels

Numerical calculations in quantum information theory are frequently prohibitively expensive, because the size of a composite state space (consisting of multiple simple systems) grows exponentially with the number of subsystems. To study composite systems consisting of more than ~ 10 qubits, using existing computers, it is important to develop efficient classical algorithms for the study of quantum information. In this Appendix, we illustrate one such algorithm, whose intended purpose is to calculate the effect of a qubit channel $\Phi^{\otimes n}$ on an n-qubit density matrix ρ. This algorithm is used to study the super-additivity of the Holevo capacity in Chapter 6. However, the application of tensor product channels to large density matrices is common in theoretical investigations into quantum information, so algorithms which perform this task more efficiently are of general interest. The remainder of this Appendix is as follows: We outline a naïve means of calculating the action of a tensor product channel on a register, which is then compared to a new algorithm, which uses the special properties of $\Phi^{\otimes n}$ under block matrix decomposition to decrease the computational time required for evaluation. We conclude by presenting source code for this algorithm which is optimized for the study of the Holevo capacity of three-state channels (see Chapter 6).

B.1 Naïve Channel Application

In order to demonstrate the efficiency of the algorithm presented in this Appendix, it is necessary to compare it with a simple, inefficient, commonly-used algorithm. In chapter

2, we introduced the Kraus and Choi representations of a quantum channel. Below, we evaluate the complexity of calculating the action of an n-qubit channel $\Phi^{\otimes n}$ in each of these representations.

The Kraus representation uses a set of operators $\{ F_j \}$ to describe a channel, the application to a density matrix ρ is

$$\Phi(\rho) = \sum_j F_j \rho F_j^\dagger. \tag{B.1}$$

For the channel $\Phi^{\otimes n}$, each of the operators is a 2^n-by-2^n matrix, as is ρ. Therefore, each matrix multiplication $F_j \rho F_j^\dagger$ requires between $\mathcal{O}(2^{2n})$ and $\mathcal{O}(2^{3n})$ operations, depending on the matrix multiplication algorithm being used [35]. The index j has r^n distinct values, where r is the rank of the Kraus map, the number of distinct operators in the set $\{ F_j \}$. We will assume, for simplicity, that $r = 4$, the maximum number of required Kraus operators for a single-qubit channel. In order to store the operators $\{ F_j \}$, then, $\mathcal{O}\left(2^{4n}\right)$ complex parameters are required, and it is necessary to perform $\mathcal{O}\left(2^{5n}\right)$ operations in order to evaluate its action on ρ.

The Choi representation uses a single 2^{2n}-by-2^{2n} matrix to store the parameters associated with a channel:

$$\beta(\Phi^{\otimes n}) = \hat{\mathbb{1}}^{\otimes n} \otimes \Phi^{\otimes n} \left(|\Omega\rangle\langle\Omega| \right), \tag{B.2}$$

$$\text{where } |\Omega\rangle = \frac{1}{2^n} \sum_{k \in \{ 0,1 \}^n} |k\rangle \otimes |k\rangle \tag{B.3}$$

To evaluate the action of a channel on a density matrix ρ in the Choi representation, we exploit the fact that the Choi matrix can be re-written in terms of the application of Φ on elementary matrices $|j\rangle\langle k|$, with j, k being bitstrings from the set $\{ 0, 1 \}^n$:

$$\beta(\Phi) = \frac{1}{2^n} \begin{bmatrix} \Phi(|0\rangle\langle 0|^{\otimes n}) & \cdots & \Phi(|0\rangle\langle 1|^{\otimes n}) \\ \vdots & \Phi(|j\rangle\langle k|) & \vdots \\ \Phi(|1\rangle\langle 0|^{\otimes n}) & \cdots & \Phi(|1\rangle\langle 1|^{\otimes n}) \end{bmatrix}. \tag{B.4}$$

Using this fact, we can write the evaluation of $\Phi(\rho)$ as

$$\rho_{a,b} \mapsto \sum_{j,k} \rho_{j,k} \left(\Phi(|j\rangle\langle k|)_{a,b} \right), \ a, b, j, k \in \{ 0, 1 \}^n. \tag{B.5}$$

A single multiplication operation is associated with each value of a, b, j, k, for a total of $\mathcal{O}\left(2^{4n}\right)$ operations required for evaluation. This is more efficient than the Kraus representation, due to the absence of matrix multiplication. The amount of storage required for the channel is identical to that required to store the Kraus representation, $\mathcal{O}\left(2^{4n}\right)$ parameters.

For a general n-qubit channel, these naïve algorithms are as efficient as possible. The tensor product channels introduced earlier have additional structure, in that they can be decomposed into a series of 1-qubit channels acting on n-qubit registers. These simple channels also have simple block decompositions, which we will exploit, in the following section, to evaluate the action of a tensor product channel efficiently.

B.2 Efficient Application of Tensor Product Channels

To derive an efficient algorithm to apply tensor product channels, we first express the action of such a channel as

$$\Phi^{\otimes n} = \bigcirc_{j=1}^{n} \Phi_j, \tag{B.6}$$

where $\Phi \circ \Lambda(\rho) = \Phi(\Lambda(\rho))$, $\bigcirc_{j=1}^{n} \Phi_j = \Phi_1 \circ \Phi_2 \circ \cdots \circ \Phi_n$, and $\Phi_j = \hat{\mathbb{1}}^{\otimes j-1} \otimes \Phi \otimes \hat{\mathbb{1}}^{\otimes n-j}$, In order to evaluate the action of the composition $\bigcirc_{j=1}^{n} \Phi_j$, we need only evaluate the action of Φ_j for all values of j from 1 to n. There exist two special cases in which a simple block decomposition can be used to efficiently evaluate Φ_j. These are the cases in which $j = 1$ and $j = n$. We describe these cases, then combine them to provide an equally-efficient method for implementing an arbitrary Φ_j.

Consider first the Kraus representation of $\Phi_1 = \Phi \otimes \hat{\mathbb{1}}^{\otimes n-1}$:

$$\Phi_1(\rho) = \sum_j F_j \otimes \hat{\mathbb{1}}^{\otimes n-1} \cdot \rho \cdot F_j^\dagger \otimes \hat{\mathbb{1}}^{\otimes n-1} \tag{B.7}$$

Each tensor product of the form $F_j \otimes \hat{\mathbb{1}}^{\otimes n-1}$ has a convenient block decomposition:

$$F_j \otimes \hat{\mathbb{1}}^{\otimes n-1} = \left[\begin{array}{c|c} F_{j,00}\hat{\mathbb{1}}^{\otimes n-1} & F_{j,01}\hat{\mathbb{1}}^{\otimes n-1} \\ \hline F_{j,10}\hat{\mathbb{1}}^{\otimes n-1} & F_{j,11}\hat{\mathbb{1}}^{\otimes n-1} \end{array}\right] \tag{B.8}$$

The product $F_j \otimes \hat{\mathbb{1}}^{\otimes n-1} \cdot \rho \cdot F_j^\dagger \otimes \hat{\mathbb{1}}^{\otimes n-1}$ can then be calculated by block-multiplying matrices:

$$\left[\begin{array}{c|c} F_{j,00}\hat{\mathbb{1}}^{\otimes n-1} & F_{j,01}\hat{\mathbb{1}}^{\otimes n-1} \\ \hline F_{j,10}\hat{\mathbb{1}}^{\otimes n-1} & F_{j,11}\hat{\mathbb{1}}^{\otimes n-1} \end{array}\right] \left[\begin{array}{c|c} \rho_{\{0,0\}} & \rho_{\{0,1\}} \\ \hline \rho_{\{1,0\}} & \rho_{\{1,1\}} \end{array}\right] \left[\begin{array}{c|c} F_{j,00}^*\hat{\mathbb{1}}^{\otimes n-1} & F_{j,10}^*\hat{\mathbb{1}}^{\otimes n-1} \\ \hline F_{j,01}^*\hat{\mathbb{1}}^{\otimes n-1} & F_{j,11}^*\hat{\mathbb{1}}^{\otimes n-1} \end{array}\right] = \left[\begin{array}{c|c} \tilde{\rho}_{\{0,0\}} & \tilde{\rho}_{\{0,1\}} \\ \hline \tilde{\rho}_{\{1,0\}} & \tilde{\rho}_{\{1,1\}} \end{array}\right] \tag{B.9}$$

where $\tilde{\rho}_{\{a,b\}} = \sum_{c,d} F_{j,ac} \cdot F_{j,bd}^* \cdot \rho_{\{c,d\}}$. Here, the $\mathcal{O}\left(2^{3n}\right)$ operations necessary to naïvely evaluate a matrix product have been replaced by $\mathcal{O}\left(2^{2n}\right)$ operations.

A similar decrease in the number of required operations is seen for Φ_n. The block decomposition of the Kraus operator $\hat{\mathbb{1}}^{\otimes n-1} \otimes F_j$ yields a block-diagonal matrix:

$$\hat{\mathbb{1}}^{\otimes n-1} \otimes F_j = \left[\begin{array}{c|c|c} F_j & \cdots & 0 \\ \hline \vdots & \ddots & \vdots \\ \hline 0 & \cdots & F_j \end{array}\right]$$

$$\hat{\mathbb{1}}^{\otimes n-1} \otimes F_j \cdot \rho \cdot \hat{\mathbb{1}}^{\otimes n-1} \otimes F_j^\dagger = \left[\begin{array}{c|c|c} F_j\rho_{\{0,0\}}F_j^\dagger & \cdots & F_j\rho_{\{0,2^{n-1}-1\}}F_j^\dagger \\ \hline \vdots & F_j\rho_{\{k,l\}}F_j^\dagger & \vdots \\ \hline F_j\rho_{\{2^{n-1}-1,0\}}F_j^\dagger & \cdots & F_j\rho_{\{2^{n-1}-1,2^{n-1}-1\}}F_j^\dagger \end{array}\right] \tag{B.10}$$

The block decomposition above replaces a single matrix multiplication of three 2^n-by-2^n matrices with $\mathcal{O}\left(2^{2n}\right)$ multiplications of 2-by-2 matrices, ensuring that each operator in the Kraus map can be evaluated in $\mathcal{O}\left(2^{2n}\right)$ operations.

To see how these two special cases can be combined in order to evaluate $\Phi_j(\rho)$ for any j, we first note that the formulae above, which are derived specifically for the Kraus representation, can be abstracted to channels in no fixed representation:

$$\hat{\mathbb{1}}^{\otimes n-1} \otimes \Phi(\rho) = \left[\begin{array}{c|c|c} \Phi(\rho_{\{0,0\}}) & \cdots & \Phi(\rho_{\{0,2^{n-1}-1\}}) \\ \hline \vdots & \Phi(\rho_{\{k,l\}}) & \vdots \\ \hline \Phi(\rho_{\{2^{n-1}-1,0\}}) & \cdots & \Phi(\rho_{\{2^{n-1}-1,2^{n-1}-1\}}) \end{array}\right] \tag{B.11}$$

$$\Phi \otimes \hat{\mathbb{1}}^{\otimes n-1}(\rho) = \Phi\left(\left[\begin{array}{c|c} \rho_{\{0,0\}} & \rho_{\{0,1\}} \\ \hline \rho_{\{1,0\}} & \rho_{\{1,1\}} \end{array}\right]\right) \tag{B.12}$$

The evaluation of the action of Φ_j is equivalent to evaluating the action of $\Phi \otimes \hat{\mathbb{1}}^{\otimes n-j}$ on the 2^{j-1}-by-2^{j-1} blocks of ρ, which can be accomplished in two steps. First, the matrix ρ is partitioned into 2^{j-1}-by-2^{j-1} blocks, then the channel Φ is applied to these blocks as 2-by-2 block matrices, each sub-block being 2^{j-2}-by-2^{j-2}. In this way, $\Phi_j(\rho)$ can be evaluated using $\mathcal{O}\left(2^{2n}\right)$ operations.

Given the abstract descriptions of channel action given by Equations B.11 and B.12, it is possible to execute this algorithm for channels in arbitrary representations of a given channel, as needed. In the following section, we present an implementation of this algorithm in MATLAB.

B.3 Implementation

The code below is presented here for completeness. Given the discussion of the algorithm above, the code below can be translated into any programming language. MATLAB is used here, due to its familiarity for a scientific audience. Also, the specific code presented here is intended for the study of the Holevo capacity in Chapter 6, uses the Choi matrix to implement $\Phi \otimes \hat{\mathbb{1}}^{\otimes n-j}$, and uses a flag to select whether the dual channel is used.

```
function [ rho ] = apply_channel_to_all( channel, rho, dualFlag )
%#codegen
%apply_channel_to_all yields the result of applying the Kraus channel
% "channel" to the all qubits of the density matrix "rho_in".
for idxQbit = 1:log2(length(rho))
    rho=apply_channel_to_one( channel, rho, uint32(idxQbit), uint8(dualFlag) );
end
end


function [ rho ] = apply_channel_to_one( channel, rho, qubit, dualFlag )

%apply_channel_to_one yields the result of applying the Kraus channel
% "channel" to the qubit "qubit" of the density matrix "rho_in".

%% Kraus-based code

%rhoOut=complex(zeros(size(rhoIn)),zeros(size(rhoIn)));
%[~,~,leng]=size(channel);
%for idxOp = 1:leng
%    rhoOut=rhoOut+apply_op_one_qubit( channel(:,:,idxOp), rhoIn, qubit );
%end

%% Choi-based code -- assumes three-state channel with special Choi matrix:
%  Here, channel is an array consisting of
%  (t_3)/2, (lambda_1)/2, (lambda_2)/2, and (lambda_3)/2:

%% Temporary hack for typecasting input dualFlag
dualFlag = uint8(dualFlag);

%% Dimensions, determining whether to apply spmd
block_size=length(rho)/(2^(qubit-1));

for idx_block_row=1:2^(qubit-1) %Loop over blocks horizontally
    for idx_block_col=1:2^(qubit-1) %Loop over blocks vertically
```

```
        %Indexing below is messy, but it just takes an appropriately sized
        %block of rhoIn and applies the one-qubit operator to the top
        %qubit.
        rho((idx_block_row-1)*block_size+1:idx_block_row*block_size,...
            (idx_block_col-1)*block_size+1:idx_block_col*block_size)=...
            top_apply_choi(channel,...
            rho((idx_block_row-1)*block_size+1:idx_block_row*block_size,...
            (idx_block_col-1)*block_size+1:idx_block_col*block_size),dualFlag);
    end
end

end


function [ rho ] = top_apply_choi( channel, rho, dualFlag )
%#codegen
%top_apply_choi uses an extremely special form for three-state channels to
%evaluate the action of a one-qubit channel on the top qubit in a register.

%% First, split rho_in into the four half-size sub-blocks:

rho_size=length(rho);
rho11=rho(1:rho_size/2,1:rho_size/2);
rho12=rho(1:rho_size/2,rho_size/2+1:rho_size);
rho21=rho(rho_size/2+1:rho_size,1:rho_size/2);
rho22=rho(rho_size/2+1:rho_size,rho_size/2+1:rho_size);

%% Pre-calculate multiplications (4-ish are required)
if dualFlag==0

    diagonal_sum=(rho11+rho22);
    half_diagonal_sum=diagonal_sum/2;
    scaled_diagonal_sum=channel(1)*diagonal_sum;
    scaled_diagonal_diff=channel(4)*(rho11-rho22);
    plus_term=channel(2)*(rho12+rho21);
    minus_term=channel(3)*(rho12-rho21);

    rho=[half_diagonal_sum+scaled_diagonal_sum+scaled_diagonal_diff,...
     plus_term + minus_term; plus_term - minus_term,...
      half_diagonal_sum-scaled_diagonal_sum-scaled_diagonal_diff];

elseif dualFlag==1

    plus_z=channel(1)+channel(4);
    minus_z=channel(1)-channel(4);
```

```
    diagonal_sum=rho11+rho22;
    diagonal_diff=rho11-rho22;
    off_diagonal_sum=rho12+rho21;
    off_diagonal_diff=rho12-rho21;
    out_off_diag_x=channel(2)*off_diagonal_sum;
    out_off_diag_y=channel(3)*off_diagonal_diff;
    top_left=out_off_diag_x + out_off_diag_y;
    bottom_right=out_off_diag_x - out_off_diag_y;

    rho=[diagonal_sum/2 + plus_z*diagonal_diff,  top_left;
        bottom_right, diagonal_sum/2+minus_z*diagonal_diff];

end
end
```

Appendix C

QuaEC: Quantum Error Correction in Python

In order to search for Clifford operators which perform fault-tolerant transcoding operations (see Chapter 7), it was necessary to develop a code library for defining elements of the Pauli and Clifford groups, as well as stabilizer codes and related quantum circuits. This library is called QuaEC (for Quantum Error Correction), it is written in Python [58], and it is publicly available [19]. This library, however, is not only useful for examining the problem of converting data between quantum codes, but also for generic Gottesman-Knill computations [60], solving commutation constraints among n-qubit Pauli operators, and expressing Clifford operators in terms of their action on the Pauli group, as unitary matrices, and as quantum circuits. These tasks are commonplace in the study of quantum computing, so it is worthwhile to automate them. The goal of this appendix is to familiarize the reader with QuaEC, explaining the principles which inform its design, and giving examples of its use. The interested reader will likely be able, after reading this appendix and the references herein, to replicate the core functionality of QuaEC in any programming language. In the following, a degree of familiarity with Python is assumed, novice readers should refer to [58] for a comprehensive introduction.

We begin by detailing the representations of Pauli and Clifford operators, along with Pauli/Clifford arithmetic (multiplication of Paulis and Cliffords and conjugation of Paulis by Cliffords). We conclude by discussing one of the more complicated problems that QuaEC is designed to solve, the derivation of sets of Paulis which satisfy arbitrary commutation/anti-commutation constraints. Note that we omit discussion of many of the high-level functions within QuaEC (for a complete description of QuaEC's capabilities, see [19]). We focus here on the solution of commutation constraints, because it is a frequently-encountered

problem in quantum information, and because the algorithm used to solve it is original (algorithms which QuaEC implements which are not original are referenced in [19]).

C.0.1 Aside: Notes on Terminology and Obtaining/Using QuaEC

There are two important points that bear mention, in order to facilitate the understanding of this Appendix, and of the QuaEC library itself. The first is the terminology of object-oriented programming, the programming paradigm under which QuaEC is designed. The second is the means by which one can obtain the necessary software to begin using QuaEC. Readers who are familiar with object-oriented programming and the Python programming language may advance to the following section immediately.

Object-oriented programming is a set of design practices used in computer programming when studying data structures with defined attributes. Consider, as a simple example, the rational numbers. Each rational number can be expressed as a/b, where a and b are integers. A rational number can be defined as an object, with integer-valued attributes (commonly called *properties*) a and b. A set of objects forms a *class*, which is defined by the permissible values of the properties.

Operations on objects are referred to as *methods*, they are similar to functions, designed to take the properties of an object as input. Methods are used to alter the properties of an object, or produce some output which depends on the object in question. Again considering the example of the rational numbers, we can define an inversion method, which performs the map $a \leftrightarrow b$, providing the multiplicative inverse of the rational number defined by a and b. We can also define a method which prints rational numbers in the form `a/b`. There also exist *static methods*, which produce instances of the class to which they are associated, without input drawn from that class. In the case of the rational numbers, we can define a method which takes a floating-point number as input, producing a rational approximation as output, for example. In addition, we can assign specific methods to *operators* (special characters such as `+` or `*` in a given programming language; note that this definition of an operator is distinct from the definition of a Pauli or Clifford operator), a process called *operator overloading*. The terms introduced here (objects, classes, methods, static methods, operators, operator overloading) will be used to describe the approach taken to representing Pauli and Clifford group elements, and performing calculations on these operators.

The second necessity for the understanding and use of this Appendix is that the reader have a Python interpreter and the QuaEC library installed, so that the reader can run the examples within this Appendix, and alter them to further examine them. Throughout the

remainder of this Appendix, examples which are intended to be run in the interpreter have lines beginning in '>>>'. The indented lines below show the expected output. There are also samples of QuaEC's internal code distributed throughout the Appendix; they do not contain the prompt character >>>.

Though QuaEC will function in any Python interpreter, we recommend IPython, an interactive interpreter with many advanced features. In addition, we recommend installation of QuaEC directly from the shell, running the following command in any operating system, once Python is installed:

```
>>> easy_install quaec
```

The above introduction to object-oriented programming and Python is meant to be sufficient for using and understanding the examples in this Appendix; for a more in-depth treatment of object-oriented programming in Python, see [58], Part VI. We resume the discussion of QuaEC's features and use below.

C.1 Representations of the Pauli and Clifford Groups

In order to perform calculations using elements of the Pauli and Clifford groups, it is first necessary to store them in a form which is both amenable to computation and interpretation by the user. In this section, we describe how QuaEC stores and manipulates Paulis and Cliffords, focusing first on the ways in which these operators can be constructed, then describing the implementation of Pauli/Clifford arithmetic in QuaEC.

C.1.1 Representation

We represent a Pauli as the product of a global phase $i^{\texttt{ph}}$, where `ph` is an integer, with a string `op`. In the example below, we initialize a Pauli operator $iX \otimes Y$:

```
>>> #import module containing error correction library, assigning local name q
>>> import qecc as q
>>> #create Pauli operator using class from q
>>> demo_pauli = q.Pauli("XY",9)
>>> demo_pauli
    i^1 XY
>>> demo_pauli.op
```

```
    "XY"
>>> demo_pauli.ph
    1
```

Note that the phase is stored modulo 4, since $i^{\texttt{ph}+4} = i^{\texttt{ph}}$.

For convenience, there are two other means of initializing a Pauli operator. For Pauli operators of low weight (those which differ from the identity on only a few qubits), we implement a static method `from_sparse` in the `Pauli` class, seen below. (In the code example below, and those that follow, we will assume that `import qecc as q` has been performed.)

```
>>> from qecc import X, Y, Z # Static methods for single-qubit Paulis
>>> #Create 20-qubit Pauli with weight 3
>>> #with operator Z on qubit 2, Y on qubit 10, X on qubit 19.
>>> sparse_pauli = q.Pauli.from_sparse({2:Z, 10:Y, 19:X}, nq=20)
>>> sparse_pauli
    i^0 IIZIIIIIIIYIIIIIIIX
```

Also, for Paulis which, on a given qubit, are either I or P, where $P \in \{ X, Y, Z \}$, the method `from_string` can be used:

```
>>> from qecc import X, Y, Z # Static methods for single-qubit Paulis
>>> bitstring = "10101101100"
>>> #Replace all "1"s with Pauli Xs, "0"s with identities.
>>> q.Pauli.from_string(bitstring,X)
    i^0 XIXIXXIXXII
>>> #As above, substituting "Y" for "X".
>>> q.Pauli.from_string(bitstring,Y)
    i^0 YIYIYYIYYII
>>> #As above, with "Z".
>>> q.Pauli.from_string(bitstring,Z)
    i^0 ZIZIZZIZZII
```

These methods are often used elsewhere within QuaEC to define stabilizer group elements, Pauli errors on registers, etc. Users are not limited to using these methods in isolation, however. Using the general initializer for the Pauli class, it is simple to define new functions and methods which return Pauli objects, by specifying `op` and `ph` for the output Pauli.

Initializing a Clifford operator is a similarly simple process. In Chapter 3, we noted that a Clifford can be specified by its action on the generating set of Paulis $\{ X_j, Z_j \,|\, 1 \leq j \leq n \}$, which we will denote $\{ \bar{X}_j, \bar{Z}_j \,|\, 1 \leq j \leq n \}$. In QuaEC, these two lists are referred to

internally as `xbars` and `zbars`, and assignments to these lists are used to initialize Clifford objects:

```
>>> #Succinctly specify lists of Paulis using the PauliList class
>>> demo_xbars = q.PauliList("XX", "ZZ");
>>> demo_zbars = q.PauliList("ZI", "IX")
>>> #Initialize a Clfiford which maps XI, IX
>>> #to demo_xbars, and ZI, IZ to demo_zbars
>>> demo_clifford = q.Clifford(demo_xbars, demo_zbars)
>>> print demo_clifford
    XI |—>  +XX
    IX |—>  +ZZ
    ZI |—>  +ZI
    IZ |—>  +IX
```

Note that any list of Pauli operators can be used to specify a Clifford, even those that do not obey the appropriate commutation constraints ($\left[\bar{X}_j, \bar{Z}_k\right] = 0$ when $j \neq k$, $\left\{\bar{X}_j, \bar{Z}_k\right\} = 0$ when $j = k$, and $\left[\bar{X}_j, \bar{X}_k\right] = \left[\bar{Z}_j, \bar{Z}_k\right] = 0$, $\forall j, k$). No warning or error is printed:

```
>>> #Same example as above, with anti—commuting demo_xbars,
>>> #and demo_zbars[1] commuting with demo_xbars[1].
>>> demo_xbars = q.PauliList("XX", "YI"); demo_zbars = q.PauliList("ZI", "IX")
>>> demo_clifford = q.Clifford(demo_xbars, demo_zbars)
>>> print demo_clifford
    XI |—>  +XX
    IX |—>  +YI
    ZI |—>  +ZI
    IZ |—>  +IX
```

To check whether the outputs of a Clifford obey the commutation constraints, we use the method `is_valid` on the Clifford itself:

```
>>> demo_xbars_good = q.PauliList("XX", "ZZ")
>>> demo_xbars_bad = q.PauliList("XX", "YI")
>>> demo_zbars = q.PauliList("ZI", "IX")
>>> demo_clifford_good = q.Clifford(demo_xbars_good, demo_zbars)
>>> demo_clifford_bad = q.Clifford(demo_xbars_bad, demo_zbars)
>>> #Check that commutation/anti—commutation constraints are satisfied.
>>> demo_clifford_good.is_valid()
    True
>>> demo_clifford_bad.is_valid()
    False
```

Checking these commutation constraints uses an amount of work which scales quadratically with respect to n (the number of qubits in question), so it is not performed on initialization. It is, however, easy for the user (or a function) to check after initialization. The class implementing Clifford operators, then, allows these objects to be initialized efficiently.

Clifford operators are often given, not in terms of their outputs $\{ \bar{X}_j, \bar{Z}_j \}$, but in terms of the generators given in Equation 3.4 in Chapter 3. QuaEC provides functions which return instances of these generators; `cnot`, `hadamard`, and `phase`. Inputs to these functions are the size of the register (`nq`) and, in the case of `hadamard` and `phase`, a single index indicating which qubit in the register is subjected to the gate. For `cnot`, two indices are used, `ctrl` and `targ`, indicating the control and target qubit indices, respectively.

```
>>> #CNOT on 2 qubits, with control on qubit 0, target on qubit 1.
>>> print q.cnot(2,0,1)
    XI |-> +XX
    IX |-> +IX
    ZI |-> +ZI
    IZ |-> +ZZ
>>> #Hadamard gate
>>> print q.hadamard(1,0)
    X |-> +Z
    Z |-> +X
>>> #Phase gate
>>> print q.phase(1,0)
    X |-> +Y
    Z |-> +Z
```

The representation detailed above is both convenient, allowing the construction of Pauli and Clifford operators in an intuitive fashion, and useful for efficient computation, since the 2^n-dimensional unitary matrices associated with a given Pauli or Clifford need not be generated in order to instantiate it. In the following subsection, we detail a similarly efficient, intuitive means of performing Pauli/Clifford products, commutations and conjugations.

C.1.2 Pauli and Clifford Arithmetic

In order to perform more complicated operations involving Pauli and Clifford operators, such as the solution of commutation constraints which concludes this chapter, it is first necessary to perform more rudimentary operations on Pauli and Clifford operators. In this chapter, we consider the multiplication of Pauli and Clifford operators, as well as determining commutation for Paulis and applying Cliffords.

Pauli Multiplication In QuaEC, the product of two single-qubit Paulis is evaluated using a 4-by-4 lookup table, stored as a dictionary:

```
MULT_TABLE = {
("I","I"): (0,"I"), ("I","X"): (0,"X"), ("I","Y"): (0,"Y"), ("I","Z"): (0,"Z"),
("X","I"): (0,"X"), ("X","X"): (0,"I"), ("X","Y"): (1,"Z"), ("X","Z"): (3,"Y"),
("Y","I"): (0,"Y"), ("Y","X"): (3,"Z"), ("Y","Y"): (0,"I"), ("Y","Z"): (1,"X"),
("Z","I"): (0,"Z"), ("Z","X"): (1,"Y"), ("Z","Y"): (3,"X"), ("Z","Z"): (0,"I")
}
```

This dictionary maps pairs of permissible values for `op` (one of `I`, `X`, `Y`, or `Z`) to tuples (lists whose contents cannot be changed, see [58]), consisting of an integer and a permissible value for `op`. The integer from the tuple, when summed with the values of `ph` for the input Paulis, gives the appropriate value of `ph` for the output. The core of the multiplication method is presented below, with `p1` and `p2` being the input multi-qubit Paulis:

```
#Initialize the output Pauli, with no entries in the operator, and a phase
#which is the product of the two input phases:
newP = Pauli("", p1.ph + p2.ph)
    #Looping over qubits, calculate new values of op and ph
    for paulicounter in range(len(p1)):
        ph, op = MULT_TABLE[(p1.op[paulicounter], p2.op[paulicounter])]
        newP.op=newP.op+op
        newP.ph=newP.ph+ph
```

The result is a Pauli which has as its `op` a concatenated string of products of single-qubit Paulis, with the sum of the resulting phases as its integer `ph`. In this manner, the product of two n-qubit Paulis can be evaluated in an amount of time that scales linearly with n. The operator `*` is overloaded to perform multiplication, for ease of use:

```
>>> p1 = q.Pauli("XYZ"); p2 = q.Pauli("ZIY", 1)
>>> p1 * p2
    i^3 YYX
```

Pauli Commutation It is also frequently necessary to determine whether two Paulis commute. For example, the determination of the validity of a Clifford operator can be reduced to determining the commutation relations of its output Paulis. Also, stabilizers and logical operators for quantum error correcting codes are defined by their commutation properties; with the stabilizers being a mutually-commuting subgroup of the Pauli group,

and the logical operators $\{ X_{L(j)}, Z_{L(j)} \mid 1 \leq j \leq k \}$ must commute with the stabilizer, and have the same commutation relations as the generators $\{ X_j, Z_j \mid 1 \leq j \leq k \}$ (see Chapter 3).

In QuaEC, Pauli commutation is determined by multiplying the input Paulis and examining the output phases:

```
#Check phases of products of Paulis P and Q
ph1, ph2 = (P*Q).ph, (Q*P).ph
#Paulis commute iff phases are equal
return 0 if ph1 == ph2 else 1
```

The output is an integer `0` (indicating commutation) or `1` (indicating anti-commutation), which can be treated as a Boolean variable. Access is provided through the function `com`:

```
>>> XYZ = q.Pauli("XYZ"); ZIY = q.Pauli("ZIY"); YYZ = q.Pauli("YYZ")
>>> #{XYZ, YYZ} = 0, [XYZ, ZIY] = 0
>>> q.com(XYZ,YYZ)
    1
>>> q.com(XYZ,ZIY)
    0
```

Since multiplication of Paulis is executed in linear time, commutation is determined in linear time as well, with a factor of two overhead, since two multiplications are required. Note that it is possible to implement a second look-up table for commutation, similar to `MULT_TABLE` which is used for multiplication above. We avoid this in the interest of legibility and portability.

Clifford Application Shifting the focus to Clifford operators, we first detail the application of a Clifford C to a Pauli P, which is the evaluation of $CPC^\dagger$. This is the fundamental step in Gottesman-Knill simulation [60]. Recall that the representation of a Clifford operator is a list of outputs from the generating set $\{ X_j, Z_j \mid 1 \leq j \leq n \}$, and that $C \left(\prod_k P_k\right) C^\dagger = \prod_k C P_k C^\dagger = \prod_k \bar{P}_k$, since C is unitary ($C^{-1} = C^\dagger$). Given the Clifford representation which QuaEC uses, then, it is possible to evaluate $CPC^\dagger$ using at most $2n$ Pauli multiplications, the application of C to the input Pauli $Y^{\otimes n}$ being the most expensive to evaluate. Since the time necessary to multiply Paulis scales linearly with n, the time necessary to apply a Clifford to a Pauli is quadratic in n.

The conjugation of Paulis by multiplication of Clifford outputs $\{ \bar{X}, \bar{Z} \}$ is carried out in the Clifford method `conjugate_pauli` using the Clifford `self` and the Pauli `pauli`:

```
for idx,op in enumerate(pauli.op):
    #For every X/Z the input Pauli contains, multiply by the
    # corresponding output Pauli from the Clifford self.
    if op == "X":
        rolling_pauli=rolling_pauli*self.xout[idx]
    elif op == "Z":
        rolling_pauli=rolling_pauli*self.zout[idx]
    elif op == "Y":
        #Y = iXZ:
        rolling_pauli=rolling_pauli*self.xout[idx]*self.zout[idx]
        rolling_pauli.mul_phase(1)
return rolling_pauli
```

This function, for every entry in the input Pauli's `op`, multiplies the output Pauli (which is initialized to the identity) by $\bar{X}_j$, $\bar{Z}_j$, or $i\bar{X}_j\bar{Z}_j$ on the right when the j^{th} entry of `op` is X, Z or $Y = iXZ$, respectively.

In QuaEC, this function can be accessed either directly, through `conjugate_pauli`, or by calling the Clifford as a function:

```
>>> #--Begin initializing Clifford--#
>>> xbars = q.PauliList("XX", "ZZ"); zbars = q.PauliList("ZI", "IX")
>>> cliff = q.Clifford(xbars, zbars)
>>> #--End initializing Clifford--#
>>> #Method on Clifford instance used below
>>> cliff.conjugate_pauli(q.Pauli("XY"))
    i^2 YZ
>>> #Cliffords can also be used as functions
>>> cliff(q.Pauli("XY"))
    i^2 YZ
>>> #For brevity, these functions can be called on strings.
>>> cliff("XY")
    i^2 YZ
```

Note that Cliffords can also be called as functions on strings; this string will be mapped to a Pauli operator with `ph = 0` if it contains only the letters `I`, `X`, `Y`, and `Z`. This compact syntax allows for the fast evaluation of the action of single Clifford operators.

Clifford Multiplication In order to evaluate the action of a series of Clifford operators, one can calculate their product. Such a product $C_2 \times C_1$ can be evaluated by repeated application of C_2 to the output Paulis of C_1, since $[C_2 \times C_1](P) = C_2(C_1(P)) \forall P$. Since C_1 is represented by its action on the generators $\{ X_j, Z_j \mid 1 \leq j \leq n \}$, the action of $C_2 \times C_1$

on these generators can be calculated using $2n$ applications of the Clifford C_2 to the Paulis $\{ C_1(X_j), C_1(Z_j) \,|\, 1 \leq j \leq n \}$.

In QuaEC, this is accomplished for two Cliffords `self` and `other`, by repeatedly calling the method `conjugate_pauli`, defined above:

```
#Loop over pairs of X and Z outputs using the Clifford self to conjugate the outputs of o
for ex, zed in zip(other.xout,other.zout):
    Xs.append(self.conjugate_pauli(ex))
    Zs.append(self.conjugate_pauli(zed))
return Clifford(Xs,Zs)
```

This functionality is accessed using the overloaded operator `*`, placed between two Cliffords:

```
>>> xbars = q.PauliList("XX","ZZ"); zbars = q.PauliList("ZI","IX")
>>> cliff = q.Clifford(xbars, zbars); other_cliff = q.Clifford(zbars, xbars)
>>> print cliff * other_cliff
    XI |->  +ZI
    IX |->  +ZZ
    ZI |->  -YY
    IZ |->  +ZX
```

Note that, in evaluating the action of a series of Cliffords on a set of Paulis, it is not always economical to calculate the product of the Cliffords in question. Suppose, for example, that there is a series of m Cliffords whose action is to be evaluated on q Paulis, all Cliffords and Paulis being defined on n qubits. Evaluating the product of m Cliffords on n qubits requires an amount of time which scales with $4 \cdot m \cdot n^3$ (each product requiring $2n$ Pauli conjugations, each requiring at most $2n$ multiplications, each multiplication requiring n operations). Application of the resulting Clifford to a set of q Paulis, then, requires an amount of time scaling with $2 \cdot q \cdot n^2$ (each application requiring $2n^2$ operations). Without multiplying the Cliffords in question, each of the m Cliffords must be applied to each of the q Paulis, with time scaling as $2 \cdot m \cdot q \cdot n^2$. When $q > \frac{mn}{m+1}$, then, less work is required to multiply the Cliffords before application to the Pauli set. If $q < \frac{mn}{m+1}$, it is less costly to apply the Cliffords in series, without multiplying them.

Composition It is possible, by composing these primitive operations, to quickly program a wide variety of tasks pertaining to Pauli and Clifford operators, making study and manipulation of these groups simple and efficient. Before advancing, in the following sections, to consider the solution of commutation constraints (one of the more complicated algorithms in QuaEC), we conclude this section with three examples of the functions detailed above.

As an example of a complex question which QuaEC can easily answer, we consider the enumeration of triples of weight-(one or two) Paulis on three qubits which are mutually anti-commuting. This example also uses `itertools`, a well-documented Python module [1], to ensure that large sets of Pauli triples are not stored in memory until requested:

```
>>> import itertools as it #Import additional module
>>> #Produce lists of weight-two Paulis on 3 qubits using:
>>> # + the QuaEC function paulis_by_weight
>>> # + the combinations function from itertools
>>> triples_of_wt_2_on_3 = it.combinations(q.paulis_by_weight(3,2),3)
>>> #Produce a Boolean function indicating whether
>>> #all the elements of the list anti-commute:
>>> list_anti_com = lambda(p_lst): \ #lambda function from Python
    all([q.com(p_lst[j], p_lst[k]) == 1 \
    for j, k in it.combinations(range(len(p_lst)),2)])
>>> #See itertools documentation re: ifilter
>>> anti_commuting_triples = it.ifilter(list_anti_com, triples_of_wt_2_on_3)
>>> #Example:
>>> list(anti_commuting_triples)[324]
    (i^0 IZI, i^0 XYI, i^0 ZYI)
```

As a further example, we examine the commutation properties of triples of Paulis drawn from partitions of the operator $XYZXYZ\ldots XYZ$. On four qubits, for instance, this is the set $\{XYZX, YZXY, ZXYZ\}$, which is mutually commuting. We can define this family of Paulis easily in QuaEC:

```
>>> #List manipulation in Python, using map:
>>> repeat_paulis = lambda nq: \
    map(q.Pauli,[("XYZ"*nq)[idx*nq:(idx+1)*nq] for idx in range(3)])
>>> #Anti-commutation and commutation constraint Boolean functions,
>>> #as above
>>> list_anti_com = lambda(p_lst): \
    all([q.com(p_lst[j], p_lst[k]) == 1 \
    for j, k in it.combinations(range(len(p_lst)),2)])
>>> list_com = lambda(p_lst): \
    all([q.com(p_lst[j], p_lst[k]) == 0 \
    for j, k in it.combinations(range(len(p_lst)),2)])
>>> #Proof that, for nq = 4,5,6,7, these lists commute for nq even,
>>> #anti-commute for nq odd.
>>> [list_com(repeat_paulis(j)) for j in range(4,8)]
    [True, False, True, False]
>>> [list_anti_com(repeat_paulis(j)) for j in range(4,8)]
```

```
    [False, True, False, True]
```

We see that these triples mutually anti-commute on odd numbers of qubits, and mutually commute on even numbers of qubits.

For a third, and final example of QuaEC's utility, we define a function that enumerates the set of errors correctable by a distance 3 CSS code (see [36, 82] for information pertaining to the codes themselves), the set $\{ X_j Z_k \mid 1 \leq j \leq n, 1 \leq k \leq n \}$:

```
import itertools as it
>>> #Express correctable set as union of:
>>> # + weight-one Paulis
>>> # + weight-two Paulis consisting of 1 "X" and 1 "Z" on different bits
>>> css_correctable_errors = lambda nq: list(q.paulis_by_weight(nq,1)) + \
    [q.Pauli.from_sparse({j:q.Z, k:q.X},nq=nq) for j,k in \
    it.permutations(range(nq),2)]
>>> #Correctable set on two qubits (only an example, no distance-3
>>> #code exists on 2 qubits).
>>> css_correctable_errors(2)
    [i^0 XI, i^0 IX, i^0 YI, i^0 IY, i^0 ZI, i^0 IZ, i^0 ZX, i^0 XZ]
```

These examples of the study of the Pauli group are a few of the ways in which QuaEC may be used, there are numerous convenience functions that have been provided in QuaEC in addition to those introduced above. For a complete list, see QuaEC's documentation [19]. These examples show some of the objects, methods, operators and functions which can be used to quickly solve various problems concerning the Pauli and Clifford groups. Having shown QuaEC's simple operations and their composition, we display one of QuaEC's more complicated features, solution of generalized commutation constraints.

C.2 Solution of Commutation Constraints

A common, complicated problem in the stabilizer formalism is the solution of commutation constraints; that is, the derivation of a set of Pauli operators $\{ P_{\text{sol}} \}$ such that every element of $\{ P_{\text{sol}} \}$ commutes with every element of a fixed set $\{ P_{\text{com}} \}$, and anti-commutes with every element of a fixed set $\{ P_{\text{acom}} \}$. This is a generalization of finding the centralizer of a set of Paulis, the set for which every element commutes with every element of $\{ P_{\text{com}} \}$. Problems of this 'generalized-centralizer' form occur in the stabilizer formalism in two circumstances:

- When analysing a Clifford operator whose action is only partially specified (suppose m of the $2n$ output Pauli generators are known), it is necessary to find Paulis which commute with $m-1$ of the specified outputs and anti-commute with the remaining Pauli.
- When determining logical operators for a stabilizer code, it is necessary to determine $2k$ Paulis which commute with the stabilizer generators, and have identical commutation relations to $\{ X_j, Z_j \mid 1 \leq j \leq k \}$.

In this section, we show the capabilities within QuaEC which have been developed to solve generalized commutation constraints. We begin with the implementation of predicates (wrappers around Boolean functions which allow for easy manipulation), a useful tool in the subsequent development of a solver for the centralizer of a Pauli set, followed by the solution of the generalized-centralizer problem introduced above.

C.2.1 Predicates

In order to represent commutation constraints in a convenient form, we use predicates, which are wrapped Boolean functions:

```
>>> #Regular Boolean function:
>>> test_pred = lambda x: x < 9
>>> test_pred(100), test_pred(1)
    (False, True)
>>> #Predicate based on the Boolean function:
>>> test_pred = q.Predicate(test_pred)
>>> test_pred(100), test_pred(1)
    (False, True)
```

What makes predicates useful is that the logical 'and/or' operations have been implemented, using `&` and `|`:

```
>>> #Create a pair of predicates:
>>> pred_high = q.Predicate(lambda x: x < 9)
>>> pred_low = q.Predicate(lambda x: x > 5)
>>> #They can be quickly composed using & and |:
>>> map(pred_high & pred_low, range(4:10)) #Logical AND of two predicates
    [False, False, True, True, True, False]
>>> map(pred_high | pred_low, range(4:10)) #Logical OR
    [True, True, True, True, True, True]
```

These operators produce `AllPredicate` and `AnyPredicate` objects, which evaluate multiple Boolean functions, returning the result of a logical `all` or `any` on the results of the component predicates (note that, for two inputs, `all` is equivalent to `and` and `any` is equivalent to `or`).

General commutation constraints can be solved using a pair of `AllPredicate` instances (one for the set with which output Paulis must commute, one for the set with which the must anti-commute); this is the preferred method in QuaEC when the Paulis satisfying the commutation/anti-commutation constraints are to be selected from a small pre-determined set:

```
if search_in_set is not None: # A Pauli search set has been specified
    commutation_predicate = AllPredicate(*map(
        lambda acc: (lambda P: pc.com(P, acc) == 0),
        commutation_constraints
        )) # AllPredicate of individual commutation predicates
    # Remove non-commuting elements from list by brute force.
    commuters = filter(commutation_predicate, search_in_set)
    anticommutation_predicate = AllPredicate(*map(
        lambda acc: (lambda P: pc.com(P, acc) == 1),
        anticommutation_constraints
        )) # Repeat for anti-commutation constraints
    # Return result of filters.
    return filter(anticommutation_predicate, commuters)
```

This method is efficient for small sets of Paulis, such as the weight-one Paulis, of which there are $3n$ on n qubits. However, for larger sets of Paulis, such as the whole Pauli group without phases (which has 4^n elements on n qubits), storing the elements in a large list and filtering them requires a prohibitively large amount of memory.

In addition, the simple method above does not take advantage of the fact that the Pauli centralizer of a set of Paulis is a subgroup of the Paulis. To see this, we note that:

- The centralizer of a set of Paulis is a subset of the Paulis, so the group operation (multiplication) is associative.
- $[PQ, R] = P[Q, R] + [P, R]Q$, so if P and Q are elements of the centralizer of a set for which R is an element, the product PQ is also in the centralizer.
- The Pauli identity $\hat{\mathbb{1}}^{\otimes n}$ trivially commutes with every Pauli, so it is in the centralizer.
- The inverse of P is either P or $-P$, both of which are in the centralizer if P is.

The group structure of the centralizer means that only its generators need be stored, so that an element of the centralizer can be derived when needed by multiplying at most $2n$ Paulis (since the entire Pauli group can be generated from $2n$ generators). Note that the anti-centralizer of a set of Paulis does not form a group, since it is not closed under multiplication (i.e. if $\{P, R\} = \{Q, R\} = 0$, then $[PQ, R] = P\{Q, R\} - \{P, R\}Q = 0$, and PQ is in the centralizer). In the following subsection, we discuss the derivation of centralizer generators, an important step in the solution of the generalized-centralizer problem.

C.2.2 Centralizer Generators

To solve the general problem of finding a set of Paulis which has mixed commutation constraints, we first narrow the search to the centralizer of those Paulis with which the solution set must commute. The centralizer of a set of Paulis is the intersection of the centralizers of the individual elements of that set. For this reason, we can first focus on finding the generators of the centralizer of a single Pauli, then use the algorithm for finding these generators as a subroutine in the derivation of centralizers of sets of Paulis.

Centralizers for Single-Qubit Paulis We suppose that we are given a single Pauli P, and a generating set for a subgroup of the Pauli group G, and we are tasked with finding the generators for the subgroup of G for which every element commutes with P. To accomplish this, QuaEC uses a recursive algorithm defined within the method `centralizer_gens` in the `Pauli` class. Given a list of group generators `group_gens` and the `Pauli self`, the recursive and base cases are evaluated in the following portion of the method:

```
if com(self, group_gens[0]) == 0:
    # That generator commutes, and so we pass it along
    # unmodified.
    return PauliList(group_gens[0], *self.centralizer_gens(group_gens[1:]))
else:
    # That generator anticommutes, and so we must modify it by
    # multiplication with another anticommuting generator, if one
    # exists.
    found = False
    for idx in range(1, len(group_gens)):
        if com(self, group_gens[idx]) == 1:
            found = True
            g_prime = group_gens[idx] * group_gens[0]
            assert com(self, g_prime) == 0
            return PauliList(g_prime, *self.centralizer_gens(group_gens[1:]))
```

```
    if not found:
        # Generator 0 anticommuted, and so we know two things
        # from our search:
        #     - All other generators commute.
        #     - The anticommuting generator (0) has no match, and must
        #       be excluded.
        return PauliList(*group_gens[1:])
```

That is, if the first of the generators we are searching over commutes with the Pauli P, we return it, along with the other generators, determined by further recursion. If, instead, the first of the generators anticommutes with P, one of two conditions holds true. If there exists another generator in the list which anti-commutes with P, we return the product of the two anti-commuting generators, along with the other generators, determined by further recursion. Conversely, if there does not exist a second anti-commuting generator in the input generators, we know that the remaining generators commute with P, so we return them, without the unpaired anti-commuting generator. Since one of these cases holds true for any Pauli in the input generators, we obtain a unique list of generators for the centralizer of a given `Pauli`.

Sets of Paulis This capability can be extended to instances of the `PauliList` class, by recursion over the list, here called `self`:

```
# Find the centralizer of the first element:
centralizer_0 = self[0].centralizer_gens(group_gens=group_gens)
# If the list has only one element, return:
if len(self) == 1:
    return centralizer_0
else:
    # The centralizer of the list must lie within the
    # centralizer of the first element, so the function recurses:
    return self[1:].centralizer_gens(group_gens=centralizer_0)
```

Since determining commutation on n-qubit Paulis requires an amount of computational time which scales linearly with n, determining the centralizer generators for a single Pauli requires an amount of work which scales at most quadratically in n, since at most $2n$ Paulis must be examined for commutation in order to determine the centralizer generators. Therefore, the amount of work required to determine the centralizer of a set of k Paulis requires an amount of work which scales as kn^2. Given this efficient subroutine, we discuss, in the next subsection, the method used by QuaEC to determine generalized centralizers.

C.2.3 Generalized Centralizers

The group properties of the centralizer are used, by the algorithm in the previous subsection, to efficiently determine and represent the centralizer of a list of Paulis in terms of a generating set. This property does not hold for the 'de-centralizer' of $\{ P_{\text{acom}} \}$ (the set of Paulis which anti-commute with every element of $\{ P_{\text{acom}} \}$), since the product of any even number of elements in the de-centralizer of $\{ P_{\text{acom}} \}$ is in the centralizer (as discussed in the beginning of this section). For this reason, QuaEC, beginning with the generators of the centralizer of $\{ P_{\text{com}} \}$, produces an *iterator* (a special list-like structure whose elements are not stored, but calculated on-demand, see [1]) onto the subset of the centralizer of $\{ P_{\text{com}} \}$ which de-centralizes $\{ P_{\text{acom}} \}$. We use this datatype because the number of elements in the centralizer scales exponentially with n, and it is frequently only necessary to derive one (for example, when searching for the logical Z in a $k = 1$ code when the stabilizer and logical X are known). This is accomplished in `solve_commutation_constraints` by defining the centralizer, then using `itertools.ifilter` (the same brute-force method employed for defined search-sets above):

```
search_in_gens =\
    commutation_constraints.centralizer_gens(group_gens=search_in_gens)

# Finally, we return a filter iterator on the elements of the given
# centralizer that selects elements which anticommute appropriately.
anticommutation_predicate = AllPredicate(*map(
    lambda acc: (lambda P: pc.com(P, acc) == 1),
    anticommutation_constraints
    ))
assert len(search_in_gens) > 0
return ifilter(anticommutation_predicate, pc.from_generators(search_in_gens))
```

C.3 Conclusions & Future Work

The development of QuaEC has resulted in a useful library of classes and functions which can be used to solve ubiquitous problems in quantum error correction. However, QuaEC can perpetually be improved, and contributions are welcome. The chief improvement that is being implemented for version 1.1 is the migration of the `Pauli` class to one which is based in the binary symplectic representation [24], a means of representing Pauli and Clifford operators using Boolean integers instead tuples consisting of integers, concatenated with

the strings `I`, `X`, `Y`, and `Z`, a form which is better-suited to user-facing tasks than intensive computation. Changing the representation used by the `Pauli` class will provide a constant-factor improvement in speed and memory-efficiency, but will require a long development process to propagate the necessary changes to the components of QuaEC which depend on the `Pauli` class. For this reason, version 1.0.1 has been released, and introduced in this appendix.

References

[1] itertools - functions creating iterators for efficient looping, 2003.

[2] Scott Aaronson and Daniel Gottesman. Improved simulation of stabilizer circuits. *Phys. Rev. A*, 70:052328, Nov 2004.

[3] P. Aliferis. *Level reduction and the quantum threshold theorem*. PhD thesis, California Institute of Technology, 2007.

[4] Panos Aliferis and Andrew W. Cross. Subsystem fault tolerance with the Bacon-Shor code. *Phys. Rev. Lett.*, 98:220502, May 2007.

[5] Panos Aliferis, Daniel Gottesman, and John Preskill. Accuracy threshold for postselected quantum computation. September 2007.

[6] D. Bacon and A. Casaccino. Quantum Error Correcting Subsystem Codes From Two Classical Linear Codes. *eprint arXiv:quant-ph/0610088*, October 2006.

[7] J. Baugh, J. Chamilliard, C. M. Chandrashekar, M. Ditty, A. Hubbard, R. Laflamme, M. Laforest, D. Maslov, O. Moussa, C. Negrevergne, M. Silva, S. Simmons, C. A. Ryan, D. G. Cory, J. S. Hodges, and C. Ramanathan. Quantum information processing using nuclear and electron magnetic resonance: review and prospects. *ArXiv e-prints*, October 2007.

[8] S. T. Belinschi, B. Collins, and I. Nechita. Almost one bit violation for the additivity of the minimum output entropy. *ArXiv e-prints*, May 2013.

[9] Charles H. Bennett, David P. DiVincenzo, John A. Smolin, and William K. Wootters. Mixed-state entanglement and quantum error correction. *Phys. Rev. A*, 54:3824–3851, Nov 1996.

[10] R.L. Boylestad and L. Nashelsky. *Electronic Devices and Circuit Theory*. Pearson-/Prentice Hall, 2009.

[11] Sergey Bravyi and Jeongwan Haah. Magic-state distillation with low overhead. *Phys. Rev. A*, 86:052329, Nov 2012.

[12] Sergey Bravyi and Alexei Kitaev. Universal quantum computation with ideal clifford gates and noisy ancillas. *Phys. Rev. A*, 71:022316, Feb 2005.

[13] H.P. Breuer and F. Petruccione. *The Theory of Open Quantum Systems*. OUP Oxford, 2007.

[14] A. R. Calderbank and Peter W. Shor. Good quantum error-correcting codes exist. *Phys. Rev. A*, 54:1098–1105, Aug 1996.

[15] Xie Chen, Hyeyoun Chung, Andrew W. Cross, Bei Zeng, and Isaac L. Chuang. Subsystem stabilizer codes cannot have a universal set of transversal gates for even one encoded qudit. *Phys. Rev. A*, 78:012353, Jul 2008.

[16] Luca Chirolli and Guido Burkard. Decoherence in solid-state qubits. *Advances in Physics*, 57(3):225–285, 2008.

[17] Man-Duen Choi. Completely positive linear maps on complex matrices. *Linear Algebra and its Applications*, 10(3):285 – 290, 1975.

[18] Claude Crépeau, Daniel Gottesman, and Adam Smith. Approximate quantum error-correcting codes and secret sharing schemes. In Ronald Cramer, editor, *Advances in Cryptology EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 285–301. Springer Berlin Heidelberg, 2005.

[19] Ben Criger and Chris Granade. Quaec: Quantum error correction in python, 2012.

[20] Ben Criger, Osama Moussa, and Raymond Laflamme. Quantum error correction with mixed ancilla qubits. *Phys. Rev. A*, 85:044302, Apr 2012.

[21] Christoph Dankert, Richard Cleve, Joseph Emerson, and Etera Livine. Exact and approximate unitary 2-designs and their application to fidelity estimation. *Phys. Rev. A*, 80:012304, Jul 2009.

[22] Nilanjana Datta and Mary Beth Ruskai. Maximal output purity and capacity for asymmetric unital qudit channels. *Journal of Physics A: Mathematical and General*, 38(45):9785, 2005.

[23] E. Davies. Information and quantum measurement. *Information Theory, IEEE Transactions on*, 24(5):596–599, 1978.

[24] Jeroen Dehaene and Bart De Moor. Clifford group, stabilizer states, and linear and quadratic operations over GF(2). *Phys. Rev. A*, 68:042318, Oct 2003.

[25] Simon J Devitt, William J Munro, and Kae Nemoto. Quantum error correction for beginners. *Reports on Progress in Physics*, 76(7):076001, 2013.

[26] David P. DiVincenzo. The physical implementation of quantum computation. *Fortschritte der Physik*, 48(9-11):771–783, 2000.

[27] David P. DiVincenzo and Peter W. Shor. Fault-tolerant error correction with efficient quantum codes. *Phys. Rev. Lett.*, 77:3260–3263, Oct 1996.

[28] Bryan Eastin and Emanuel Knill. Restrictions on transversal encoded quantum gate sets. *Phys. Rev. Lett.*, 102:110502, Mar 2009.

[29] Andrew S. Fletcher, Peter W. Shor, and Moe Z. Win. Optimum quantum error recovery using semidefinite programming. *Phys. Rev. A*, 75:012338, Jan 2007.

[30] A.S. Fletcher, P.W. Shor, and M.Z. Win. Channel-adapted quantum error correction for the amplitude damping channel. *Information Theory, IEEE Transactions on*, 54(12):5705–5718, 2008.

[31] Austin G. Fowler, Matteo Mariantoni, John M. Martinis, and Andrew N. Cleland. Surface codes: Towards practical large-scale quantum computation. *Phys. Rev. A*, 86:032324, Sep 2012.

[32] C. A. Fuchs. Nonorthogonal quantum states maximize classical information capacity. *Physical Review Letters*, 79:1162–1165, August 1997.

[33] Motohisa Fukuda, Christopher King, and David K. Moser. Comments on Hastings' additivity counterexamples. *Communications in Mathematical Physics*, 296(1):111–143, 2010.

[34] Neil A. Gershenfeld and Isaac L. Chuang. Bulk Spin-Resonance Quantum Computation. *Science*, 275(5298):350–356, 1997.

[35] G.H. Golub and C.F. Van Loan. *Matrix Computations*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, 1996.

[36] D. Gottesman. *Stabilizer codes and quantum error correction*. PhD thesis, California Institute of Technology, 1997.

[37] Daniel Gottesman and Isaac L. Chuang. Demonstrating the viability of universal quantum computation using teleportation and single-qubit operations. *Nature*, 402(6760):390–393, Nov 1999.

[38] D. M. Greenberger, M. A. Horne, and A. Zeilinger. Going Beyond Bell's Theorem. *ArXiv e-prints*, December 2007.

[39] M. B. Hastings. Superadditivity of communication capacity using entangled inputs. *Nat Phys*, 5(4):255–257, Apr 2009.

[40] T. F. Havel. Robust procedures for converting among Kraus, Lindblad and matrix representations of quantum dynamical semigroups. *J. Math. Phys.*, 44:534–557, 2003.

[41] Charles D. Hill, Austin G. Fowler, David S. Wang, and Lloyd C. L. Hollenberg. Fault-tolerant quantum error correction code conversion. *Quantum Info. Comput.*, 13(5-6):439–451, May 2013.

[42] A.S. Holevo. The capacity of the quantum channel with general signal states. *Information Theory, IEEE Transactions on*, 44(1):269–273, 1998.

[43] Michał Horodecki, Paweł Horodecki, and Ryszard Horodecki. Separability of mixed states: necessary and sufficient conditions. *Physics Letters A*, 223(1 – 2):1 – 8, 1996.

[44] Michał Horodecki, Paweł Horodecki, and Ryszard Horodecki. General teleportation channel, singlet fraction, and quasidistillation. *Phys. Rev. A*, 60:1888–1898, Sep 1999.

[45] Jonathan A. Jones. Quantum computing with NMR. *Progress in Nuclear Magnetic Resonance Spectroscopy*, 59(2):91 – 120, 2011.

[46] Phillip Kaye, Raymond Laflamme, and Michele Mosca. *An Introduction to Quantum Computing*. Oxford University Press, 2007.

[47] C. King and M. B. Ruskai. Minimal entropy of states emerging from noisy quantum channels. *eprint arXiv:quant-ph/9911079*, November 1999.

[48] Christopher King. Additivity for unital qubit channels. *J. Math. Phys*, pages 4641–4653, 2002.

[49] Christopher King, Michael Nathanson, and Mary Beth Ruskai. Qubit channels can require more than two inputs to achieve capacity. *Phys. Rev. Lett.*, 88:057901, Jan 2002.

[50] E. Knill. Quantum computing with realistically noisy devices. *Nature*, 434(7029):39–44, Mar 2005.

[51] Emanuel Knill and Raymond Laflamme. Theory of quantum error-correcting codes. *Phys. Rev. A*, 55:900–911, Feb 1997.

[52] Emanuel Knill, Raymond Laflamme, and Wojciech H. Zurek. Resilient quantum computation: error models and thresholds. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 454(1969):365–384, 1998.

[53] Yasushi Kondo, Chiara Bagnasco, and Mikio Nakahara. Implementation of a simple operator-quantum-error-correction scheme. *Phys. Rev. A*, 88:022314, Aug 2013.

[54] K. Kraus, A. Böhm, J.D. Dollard, and W.H. Wootters. *States, effects, and operations: fundamental notions of quantum theory : lectures in mathematical physics at the University of Texas at Austin*. Lecture notes in physics. Springer-Verlag, 1983.

[55] Raymond Laflamme, Cesar Miquel, Juan Pablo Paz, and Wojciech Hubert Zurek. Perfect quantum error correcting code. *Physical Review Letters*, 77(1):198, 1996.

[56] Debbie W. Leung, M. A. Nielsen, Isaac L. Chuang, and Yoshihisa Yamamoto. Approximate quantum error correction can lead to better codes. *Phys. Rev. A*, 56:2567–2573, Oct 1997.

[57] Daniel A Lidar, Isaac L Chuang, and K Birgitta Whaley. Decoherence-free subspaces for quantum computation. *Physical Review Letters*, 81(12):2594, 1998.

[58] M. Lutz. *Learning Python*. O'Reilly Media, 2013.

[59] J. Dahl M. S. Andersen and L. Vandenberghe. CVXOPT: A python package for convex optimization, version 1.1.6., 2013.

[60] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, Cambridge, UK, 2000.

[61] Masanori Ohya, Dénes Petz, and Noburu Watanabe. On capacities of quantum channels. *Prob. Math. Stat.*, 17:179–196, 1997.

[62] A. Paetznick and B. W. Reichardt. Universal fault-tolerant quantum computation with only transversal gates and error correction. *ArXiv e-prints*, April 2013.

[63] Asher Peres. Separability criterion for density matrices. *Phys. Rev. Lett.*, 77:1413–1415, Aug 1996.

[64] D. Perez-Garcia, F. Verstraete, M. M. Wolf, and J. I. Cirac. Matrix product state representations. *Quantum Info. Comput.*, 7(5):401–430, July 2007.

[65] Benjamin Rahn, Andrew C. Doherty, and Hideo Mabuchi. Exact performance of concatenated quantum codes. *Phys. Rev. A*, 66:032304, Sep 2002.

[66] Robert Raussendorf and Hans J. Briegel. A one-way quantum computer. *Phys. Rev. Lett.*, 86:5188–5191, May 2001.

[67] M. Reimpell and R. F. Werner. Iterative optimization of quantum error correcting codes. *Phys. Rev. Lett.*, 94(8):080501, Mar 2005.

[68] Michael Reimpell. *Quantum Information and Convex Optimization*. PhD thesis, Technischen Universität Carolo-Wilhelmina zu Braunschweig, 2006.

[69] M. B. Ruskai. Qubit entanglement breaking channels. *Reviews in Mathematical Physics*, 15:643–662, 2003.

[70] Mary Beth Ruskai, Stanislaw Szarek, and Elisabeth Werner. An analysis of completely-positive trace-preserving maps on M2. *Linear Algebra and its Applications*, 347(1–3):159 – 187, 2002.

[71] J. J. Sakurai. *Modern Quantum Mechanics (Revised Edition)*. Addison Wesley, 1 edition, September 1993.

[72] Benjamin Schumacher. Sending entanglement through noisy quantum channels. *Phys. Rev. A*, 54:2614–2628, Oct 1996.

[73] Benjamin Schumacher and Michael D. Westmoreland. Sending classical information via noisy quantum channels. *Phys. Rev. A*, 56:131–138, Jul 1997.

[74] C.E. Shannon. A symbolic analysis of relay and switching circuits. *American Institute of Electrical Engineers, Transactions of the*, 57(12):713–723, 1938.

[75] P. W. Shor. Additivity of the classical capacity of entanglement-breaking quantum channels. *Journal of Mathematical Physics*, 43:4334–4340, September 2002.

[76] Peter Shor. Additivity of the classical capacity of entanglement-breaking quantum channels. *J. Math. Phys*, pages 4334–4340, 2002.

[77] Peter W. Shor. Scheme for reducing decoherence in quantum computer memory. *Phys. Rev. A*, 52:R2493–R2496, Oct 1995.

[78] Peter W. Shor. Equivalence of additivity questions in quantum information theory. *Communications in Mathematical Physics*, 246(3):453–472, 2004.

[79] S. Simmons, R. M. Brown, H. Riemann, N. V. Abrosimov, P. Becker, H. J. Pohl, M. L. W. Thewalt, K. M. Itoh, and J. J. L. Morton. Entanglement in a solid-state spin ensemble. *Nature*, 470:69–72, 2011.

[80] Graeme Smith and Jon Yard. Quantum communication with zero-capacity channels. *Science*, 321(5897):1812–1815, 2008.

[81] A. M. Steane. Error correcting codes in quantum theory. *Phys. Rev. Lett.*, 77:793–797, Jul 1996.

[82] A. M. Steane. Simple quantum error-correcting codes. *Phys. Rev. A*, 54:4741–4751, Dec 1996.

[83] A. M. Steane. Active stabilization, quantum computation, and quantum state synthesis. *Phys. Rev. Lett.*, 78:2252–2255, Mar 1997.

[84] A.M. Steane. Quantum Reed-Muller codes. *Information Theory, IEEE Transactions on*, 45(5):1701–1703, 1999.

[85] Ashley M. Stephens, Zachary W. E. Evans, Simon J. Devitt, and Lloyd C. L. Hollenberg. Asymmetric quantum error correction via code conversion. *Phys. Rev. A*, 77:062335, Jun 2008.

[86] W. Forrest Stinespring. Positive functions on c*-algebras. *Proceedings of the American Mathematical Society*, 6(2):pp. 211–216, 1955.

[87] Jos F. Sturm. Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones, 1998.

[88] John Watrous. Theory of quantum information course notes. 2011.

[89] M. M. Wilde. From Classical to Quantum Shannon Theory. *ArXiv e-prints*, June 2011.

[90] Bei Zeng, A. Cross, and I.L. Chuang. Transversality versus universality for additive quantum codes. *Information Theory, IEEE Transactions on*, 57(9):6272–6284, 2011.

[91] Xinlan Zhou, Debbie W. Leung, and Isaac L. Chuang. Methodology for quantum logic gate construction. *Phys. Rev. A*, 62:052316, Oct 2000.