Stabilizer Codes and Quantum Error Correction

Thesis by
Daniel Gottesman

In Partial Fulfillment of the Requirements
for the Degree of
Doctor of Philosophy

California Institute of Technology
Pasadena, California

2024
(Submitted May 21, 1997)

Acknowledgements

I would like to thank my advisor John Preskill for his guidance, and the members of the QUIC collaboration, particularly David Beckman, John Cortese, Jarah Evslin, Chris Fuchs, Sham Kakade, Andrew Landahl, and Hideo Mabuchi, for many stimulating conversations. My graduate career was supported by a National Science Foundation Graduate Fellowship, by the U. S. Department of Energy under Grant No. DE-FG03-92-ER40701, and by DARPA under Grant No. DAAH04-96-1-0386 administered by the Army Research Office.

Abstract

Controlling operational errors and decoherence is one of the major challenges facing the field of quantum computation and other attempts to create specified many-particle entangled states. The field of quantum error correction has developed to meet this challenge. A group-theoretical structure and associated subclass of quantum codes, the stabilizer codes, has proved particularly fruitful in producing codes and in understanding the structure of both specific codes and classes of codes. I will give an overview of the field of quantum error correction and the formalism of stabilizer codes. In the context of stabilizer codes, I will discuss a number of known codes, the capacity of a quantum channel, bounds on quantum codes, and fault-tolerant quantum computation.

Contents

1 Introduction and Preliminary Material **1**
1.1 Quantum Computers 1
1.2 Introduction to Quantum Mechanics 5
1.3 Introduction to Classical Coding Theory 8

2 Basics of Quantum Error Correction **11**
2.1 The Quantum Channel 11
2.2 A Simple Code 11
2.3 Properties of Any Quantum Code 13
2.4 Error Models 15

3 Stabilizer Coding **17**
3.1 The Nine-Qubit Code Revisited 17
3.2 The General Stabilizer Code 18
3.3 Some Examples 21
3.4 Alternate Languages for Stabilizers 23
3.5 Making New Codes From Old Codes 25
3.6 Higher Dimensional States 29

4 Encoding and Decoding Stabilizer Codes **31**
4.1 Standard Form for a Stabilizer Code 31
4.2 Network for Encoding 33
4.3 Other Methods of Encoding and Decoding 35

5 Fault-Tolerant Computation **37**
5.1 Encoded Computation and Fault-Tolerance 37
5.2 Measurement and Error Correction 38
5.3 Transformations of the Stabilizer 40
5.4 The Effects of Measurements 44
5.5 Producing New Operations in $N(\mathcal{G})$ 46
5.6 Codes With Multiple Encoded Qubits 49
5.7 The Toffoli Gate 52
5.8 Construction of Gates in $N(\mathcal{G})$ 55
5.9 Refining the Error Correction Algorithm 57

6 Concatenated Coding **60**
6.1 The Structure of Concatenated Codes 60
6.2 Threshhold for Storage Errors and Gates From $N(\mathcal{G})$ 62
6.3 Toffoli Gate Threshhold 67

7 Bounds on Quantum Error-Correcting Codes **72**
7.1 General Bounds 72
7.2 Weight Enumerators and Linear Programming Bounds 74
7.3 Bounds on Degenerate Stabilizer Codes 78
7.4 Error-Correcting Codes and Entanglement Purification Protocols 81
7.5 Capacity of the Erasure Channel 82
7.6 Capacity of the Depolarizing Channel 83

8 Examples of Stabilizer Codes **87**
8.1 Distance Two Codes 87
8.2 The Five-Qubit Code 89
8.3 A Class of Distance Three Codes 90
8.4 Perfect One-Error-Correcting Codes 95
8.5 A Class of Distance Four Codes 96
8.6 CSS Codes 98
8.7 Amplitude Damping Codes 99
8.8 Some Miscellaneous Codes 101

A Quantum Gates **104**

B Glossary **106**

List of Tables

3.1 The stabilizer for Shor's nine-qubit code 17
3.2 The stabilizer for the five-qubit code. 21
3.3 The stabilizer for the eight-qubit code. 22
3.4 The seven-qubit CSS code. 23
3.5 A $[4, 2, 2]$ code derived from the $[5, 1, 3]$ code. 26
3.6 The thirteen-qubit code formed by pasting together the five- and eight-qubit codes. 27
3.7 Result of concatenating the five-qubit code with itself. 28

8.1 The stabilizer for a $[16, 10, 3]$ code. 91
8.2 The stabilizer for a $[16, 6, 4]$ code. 97
8.3 The stabilizer for the $[8, 0, 4]$ code. 98
8.4 A four-qubit code for the amplitude damping channel. 100
8.5 The stabilizer for an $[11, 1, 5]$ code. 102
8.6 The stabilizer for a code to correct one σ_x or σ_z error. 102

List of Figures

2.1 Network to detect leakage errors. 16

4.1 Creating the state $\overline{X}|00000\rangle$ for the five-qubit code. 34
4.2 Network for encoding the five-qubit code. 35

5.1 Network to swap $|\alpha\rangle$ and $|\beta\rangle$ using ancilla $|\gamma\rangle$. 42
5.2 Network to swap two qubits using CNOT. 51
5.3 Recursive construction of gates in $N(\mathcal{G})$. 57

6.1 Cat state construction and verification. 64
6.2 The Toffoli gate construction. 68

7.1 The quantum Hamming bound, the Knill-Laflamme bound, and the bound from equation (7.66) 86

A.1 Various quantum gates. 105

Chapter 1

Introduction and Preliminary Material

1.1 Quantum Computers

Computers have changed the world in many ways. They are ubiquitous, running air-traffic control and manufacturing plants, providing movie special effects and video games, and serving as a substrate for electronic mail and the World Wide Web. While computers allow us to solve many problems that were simply impossible before their advent, a number of problems require too much computation to be practical even for relatively simple inputs, and using the most powerful computers.

The field of classical complexity theory has developed to classify problems by their difficulty. A class of problems is generally considered tractable if an algorithm exists to solve it with resources (such as time and memory) polynomial in the size of the input. Two well-known classically intractable problems are factoring an n-bit number and the Traveling Salesman problem (finding the minimum cyclic path connecting n cities with specified distances between them). Both of these problems are in the complexity class NP (for "non-deterministic polynomial"):[1] given a black box that solves the problem (an *oracle*), we can check in polynomial time that the solution is correct. The Traveling Salesman problem is an NP-complete problem; that is, any problem in NP can be transformed into an instance of the Traveling Salesman problem in polynomial time. If we can solve the Traveling Salesman problem in polynomial time, we can solve any NP problem in polynomial time. Factoring may or may not be NP-complete, but so much work has been done attempting to solve it that the consensus is that it is classically intractable, and RSA public-key cryptography, which is used, for instance, to send credit-card numbers in Web browsing software, depends on the difficulty of factoring large numbers.

[1] Strictly speaking, it is the associated decision problems that are in NP.

As computer hardware develops over time, the underlying technology continually changes to become faster, smaller, and generally better. What was impossible on yesterday's computers may be quite possible today. A problem that was intractable on the earlier hardware might become tractable with the new technology. However, the strong Church-Turing Thesis [1] states that this is not the case, and that every physical implementation of universal computation can simulate any other implementation with only a polynomial slowdown.[2] In this way, the Church-Turing Thesis protects complexity theory from obsolescence as computer technology improves. While a new computer may be able to factor larger numbers, the difficulty of factoring numbers will still scale the same way with the size of the input on the new hardware as on the old hardware.

Another problem that has proven to be classically intractable is simulating quantum systems. A single spin-1/2 particle, such as an electron trapped in a quantum dot, has a two-dimensional space of states, which can be considered to describe the direction of its spin. A similar classical particle such as a Heisenberg spin would also have a two-dimensional space of states. However, n quantum particles have a 2^n-dimensional state space, while n classical Heisenberg spins would only have a $2n$-dimensional space of states. The extra states in the quantum system come from the presence of entangled states between many different particles. Note that while an n-bit classical digital computer has 2^n possible states, they only form an n-dimensional state space, since a state can be described by an n-component binary vector. To describe a state in a quantum computer with n qubits requires a complex vector with 2^n components. I give a basic introduction to quantum mechanics in section 1.2. Quantum systems are difficult to simulate classically because they generically utilize the full 2^n-dimensional Hilbert space as they evolve, requiring exponential classical resources.

This fact led Feynman to conjecture that a quantum computer which used quantum mechanics intrinsically might be more powerful than a computer mired in the classical world [2]. While this seems a sensible suggestion when just looking at quantum mechanics, it is in fact quite revolutionary in that it suggests that the strong Church-Turing Thesis is wrong![3] This opens up the possibility that classical complexity classes might not apply for quantum computers, and that some classically intractable problems might become tractable. The most spectacular instance of this is Shor's discovery of an algorithm to factor numbers on a quantum computer in a polynomial time in the number of digits [3]. Another impressive algorithm is Grover's algorithm [4], which can find a single object in an unsorted database of N objects in $O(\sqrt{N})$ time on a quantum computer, while the same task would require an exhaustive search on a classical computer, taking $O(N)$ time. It has been shown that $O(\sqrt{N})$ time is the best possible speed for this task [5], which tends to suggest that NP-complete problems are still intractable on a quantum computer, although this has not

[2] The original Church-Turing thesis only states that any universal computer can simulate any other computer, but the requirement of polynomial resources is a useful strengthening.

[3] A classical computer can simulate a quantum computer, but only with exponential resources, so the weak Church-Turing Thesis does still hold.

been shown (note that a proof of this would also show P $\neq$ NP for a classical computer).

However, declaring by theoretical fiat the basic properties of a quantum computer is a far cry from actually building one and using it to factor large numbers. Nevertheless, the first steps in building a quantum computer have been taken. Any quantum computer requires a system with long-lived quantum states and a way to interact them. Typically, we consider systems comprised of a number of two-state subsystems, which are called *qubits* (for "quantum bits"). There are many proposals for how to build a quantum computer. Some possible physical realizations of qubits are:

- the ground and excited states of ions stored in a linear ion trap, with interactions between ions provided through a joint vibrational mode [6, 7].
- photons in either polarization, with interactions via cavity QED [8].
- nuclear spin states in polymers, with interactions provided by nuclear magnetic resonance techniques [9].

While these implementations are seemingly very different, it is possible to simulate the computational process of one system on any of the others, providing a quantum analogue to the Church-Turing Thesis (although there are difficult technical or theoretical problems with scaling up the size of these implementations).

These suggested implementations of quantum computers all share a much higher susceptibility to errors than modern classical computers. While further development may reduce the size of errors by orders of magnitude, it is unlikely that quantum computers will ever reach the incredible reliability of classical computers. Modern classical computers guard against error largely by being digital instead of analog — instead of allowing each bit of the computer to vary continuously between 0 and 1, at each time step the hardware kicks the bit back to the nearer of 0 and 1. This prevents small errors from building up into large errors, which are therefore drastically reduced. The same technique cannot be used in a quantum computer, because continually measuring each qubit would destroy the entangled states that distinguish a quantum computer from a classical computer.

Entangled states are in general very delicate, and making a measurement on one will typically collapse it into a less entangled state. Small interactions with the environment provide a sort of continuous measurement of a system, and as the system grows in size, these become harder and harder to ignore. The system will *decohere* and begin to look like a classical system. Decoherence is why the world looks classical at a human scale. Reducing interactions with the environment can reduce the effects of decoherence, but not eliminate them entirely.

Even if the basal error rate in a quantum computer can be reduced to some small value ϵ per unit time, after N time steps, the probability of surviving without an error is only $(1-\epsilon)^N$, which decreases exponentially with N. Even

if an algorithm runs in polynomial time on an error-free computer, it will require exponentially many runs on a real computer unless something can be done to control the errors.

The same problem occurs for classical computers. There, the problem can be solved in principle by the use of error-correcting codes. In practice, they are not usually necessary for normal computer operation, but they are essential to overcome noise in communications channels. I give a basic introduction to the theory of classical error-correcting codes in section 1.3.

Classical error-correction techniques cannot be directly carried over to quantum computers for two reasons. First of all, the classical techniques assume we can measure all of the bits in the computer. For a quantum computer, this would destroy any entanglement between qubits. More importantly, a classical computer only needs to preserve the bit values of 0 and 1. A quantum computer also needs to keep phase information in entangled states. Thus, while quantum error-correcting codes are related to classical codes, they require a somewhat new approach.

The first quantum error-correcting codes were discovered by Shor [10] and Steane [11]. I discuss Shor's original code and some basics of quantum error-correcting codes in chapter 2. I then go on to describe the formalism of stabilizer codes in chapter 3, along with some simple examples and methods for creating new codes from old ones. Chapter 4 describes how to build networks to encode and decode stabilizer codes. Because we will want to use these codes in the operation of quantum computers, in chapter 5, I will discuss how to perform operations on states encoded using a quantum error-correcting code without losing the protection against errors. Chapter 6 describes how to use concatenated codes to do arbitrarily long calculations as long as the basic error rate is below some threshhold value, and presents a rough calculation of that threshhold. Chapter 7 discusses known upper and lower bounds on the existence of stabilizer codes and the channel capacity. Finally, in chapter 8, I will give a partial list of known quantum error-correcting codes and their properties. Appendix A contains a brief discussion of quantum gates and a list of symbols for them used in figures. Appendix B contains a glossary of useful terms for discussing quantum error-correcting codes.

Since the promise of quantum computation has attracted scientists from a number of fields, including computer science, mathematics, and physics, some of the background one group takes for granted may be alien to others. Therefore, in the following two sections, I have provided basic introductions to quantum mechanics and classical coding theory. People familiar with one or both fields should skip the appropriate section(s). For a more complete treatment of quantum mechanics, see [12]. For a more complete treatment of classical error-correcting codes, see [13].

1.2 Introduction to Quantum Mechanics

The state of a classical computer is a string of 0s and 1s, which is a vector over the finite field $\mathbf{Z}_2$. The state of a quantum computer (or any quantum system) is instead a vector over the complex numbers $\mathbf{C}$. Actually, a quantum state lies in a Hilbert space, since there is an inner product (which I will define later). The state is usually written $|\psi\rangle$, which is called a *ket*. A classical computer with n bits has 2^n possible states, but this is only an n-dimensional vector space over $\mathbf{Z}_2$. A quantum computer with n qubits is a state in a 2^n-dimensional complex vector space. For a single qubit, the standard basis vectors are written as $|0\rangle$ and $|1\rangle$. An arbitrary single-qubit state is then

$$\alpha|0\rangle + \beta|1\rangle. \tag{1.1}$$

α and β are complex numbers, with $|\alpha|^2 + |\beta|^2 = 1$. This is a *normalized* state. With multiple qubits, we can have states that cannot be written as the product of single-qubit states. For instance,

$$\frac{1}{\sqrt{2}}\left(|00\rangle + |11\rangle\right) \tag{1.2}$$

cannot be decomposed in this way. Such a state is said to be *entangled*. Entangled states are what provide a quantum computer with its power. They will also play a major role in quantum error correction. The particular state (1.2) is called an Einstein-Podalsky-Rosen pair (or EPR) pair, and serves as a useful basic unit of entanglement in many applications.

If we make a measurement on the qubit in equation (1.1), we get a classical number corresponding to one of the basis states. The measurement disturbs the original state, which collapses into the basis state corresponding to the measurement outcome. If we measure the state (1.1), the outcome will be 0 with probability $|\alpha|^2$, and it will be 1 with probability $|\beta|^2$. The normalization ensures that the probability of getting some result is exactly 1. Through most of this thesis, I will instead write down unnormalized states. These states will stand for the corresponding normalized states, which are formed by multiplying the unnormalized states by an appropriate constant. The overall phase of a state vector has no physical significance.

The measurement we made implements one of two projection operators, the projections on the basis $|0\rangle$, $|1\rangle$. This is not the only measurement we can make on a single qubit. In fact, we can project on any basis for the Hilbert space of the qubit. If we have multiple qubits, we can measure a number of different qubits independently, or we can measure some joint property of the qubits, which corresponds to projecting on some entangled basis of the system. Note that the projection on the basis $|0\rangle$, $|1\rangle$ for either qubit destroys the entanglement of the state (1.2), leaving it in a tensor product state.

A particularly fruitful way to understand a quantum system is to look at the behavior of various operators acting on the states of the system. For instance, a nice set of operators to consider for a single qubit is the set of Pauli spin

matrices

$$\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \ \sigma_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \text{ and } \sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{1.3}$$

The original measurement I described corresponds to measuring the eigenvalue of σ_z. The corresponding projection operators are $\frac{1}{2}(I \pm \sigma_z)$. If we have a spin-1/2 particle, this measurement is performed by measuring the spin of the particle along the z axis. We could also measure along the x or y axis, which corresponds to measuring the eigenvalue of σ_x or σ_y. The projections are $\frac{1}{2}(I \pm \sigma_x)$ and $\frac{1}{2}(I \pm \sigma_y)$.

We can also make measurements of more general operators, provided they have real eigenvalues. A matrix A has real eigenvalues iff it is Hermitian: $A^\dagger = A$, where $A^\dagger$ is the *Hermitian adjoint* (or just *adjoint*), equal to the complex conjugate transpose. Note that all of the Pauli spin matrices are Hermitian.

The Pauli matrices also satisfy an important algebraic property — they *anticommute* with each other. That is,

$$\{\sigma_i, \sigma_j\} = \sigma_i\sigma_j + \sigma_j\sigma_i = 0 \tag{1.4}$$

whenever $i \neq j$ (with $i, j \in \{x, y, z\}$). Another possible relationship between two operators A and B is for them to *commute*. That is,

$$[A, B] = AB - BA = 0. \tag{1.5}$$

It is possible for two matrices to neither commute nor anticommute, and, in fact, this is the generic case. Two commuting matrices can be simultaneously diagonalized. This means that we can measure the eigenvalue of one of them without disturbing the eigenvectors of the other. Conversely, if two operators do not commute, measuring one will disturb the eigenvectors of the other, so we cannot simultaneously measure non-commuting operators.

There is a natural complex inner product on quantum states. Given an orthonormal basis $|\psi_i\rangle$, the inner product between $|\alpha\rangle = \sum c_i|\psi_i\rangle$ and $|\beta\rangle = \sum d_i|\psi_i\rangle$ is

$$\langle\alpha|\beta\rangle = \sum c_i^* d_j \langle\psi_i|\psi_j\rangle = \sum c_i^* d_i. \tag{1.6}$$

Each ket $|\psi\rangle$ corresponds to a *bra* $\langle\psi|$ and the Hermitian adjoint is the adjoint with respect to this inner product, so $U|\psi\rangle$ corresponds to $\langle\psi|U^\dagger$. The operator $\sum |\psi\rangle\langle\phi|$ acts on the Hilbert space as follows:

$$\left(\sum |\psi\rangle\langle\phi|\right)|\alpha\rangle = \sum \langle\phi|\alpha\rangle \ |\psi\rangle. \tag{1.7}$$

The inner product can reveal a great deal of information about the structure of a set of states. For instance, $\langle\psi|\phi\rangle = 1$ if and only if $|\psi\rangle = |\phi\rangle$.

Eigenvectors of a Hermitian operator A with different eigenvalues are automatically orthogonal:

$$\langle\psi|A|\phi\rangle = \langle\psi|\big(A|\phi\rangle\big) = \lambda_\phi\langle\psi|\phi\rangle \tag{1.8}$$

$$= \big(\langle\psi|A\big)|\phi\rangle = \lambda_\psi^*\langle\psi|\phi\rangle. \tag{1.9}$$

Since the eigenvalues of A are real, it follows that $\langle\psi|\phi\rangle = 0$ whenever $\lambda_\phi \neq \lambda_\psi$. Conversely, if $\langle\psi|\phi\rangle = 0$, there exists a Hermitian operator for which $|\psi\rangle$ and $|\phi\rangle$ are eigenvectors with different eigenvalues.

We often want to consider a subsystem $\mathcal{A}$ of a quantum system $\mathcal{B}$. Since $\mathcal{A}$ may be entangled with the rest of the system, it is not meaningful to speak of the "state" of $\mathcal{A}$. If we write the state of $\mathcal{B}$ as $\sum |\psi_i\rangle|\phi_i\rangle$, where $|\psi_i\rangle$ is an orthonormal basis for $\mathcal{B} - \mathcal{A}$, and $|\phi_i\rangle$ are possible states for $\mathcal{A}$, then to an observer who only interacts with the subsystem $\mathcal{A}$, the subsystem appears to be in just one of the states $|\phi_i\rangle$ with some probability. $\mathcal{A}$ is said to be in a *mixed state* as opposed to the *pure state* of a closed system in a definite state.

We can extend the formalism to cover mixed states by introducing the *density matrix* ρ. For a pure system in the state $|\psi\rangle$, the density matrix is $|\psi\rangle\langle\psi|$. The density matrix for the subsystem for the entangled state above is $\sum |\phi_i\rangle\langle\phi_i|$. Density matrices are always positive and have $\operatorname{tr}\rho = 1$. To find the density matrix of a subsystem given the density matrix of the full system, simply trace over the degrees of freedom of the rest of the system.

Given a closed quantum system, time evolution preserves the inner product, so the time evolution operator U must be unitary. That is, $U^\dagger U = UU^\dagger = I$. An open system can be described as a subsystem of a larger closed system, so the evolution of the open system descends from the global evolution of the full system. Time evolution of the subsystem is described by some *superoperator* acting on the density matrix of the subsystem.

One fact about quantum states that has profound implications for quantum computation is that it is impossible to make a copy of an arbitrary unknown quantum state. This is known as the "No Cloning Theorem," [14] and is a consequence of the linearity of quantum mechanics. The proof is straightforward: Suppose we wish to have an operation that maps an arbitrary state

$$|\psi\rangle \to |\psi\rangle \otimes |\psi\rangle. \tag{1.10}$$

Then arbitrary $|\phi\rangle$ is mapped by

$$|\phi\rangle \to |\phi\rangle \otimes |\phi\rangle \tag{1.11}$$

as well. Because the transformation must be linear, it follows that

$$|\psi\rangle + |\phi\rangle \to |\psi\rangle \otimes |\psi\rangle + |\phi\rangle \otimes |\phi\rangle. \tag{1.12}$$

However,

$$|\psi\rangle \otimes |\psi\rangle + |\phi\rangle \otimes |\phi\rangle \neq (|\psi\rangle + |\phi\rangle) \otimes (|\psi\rangle + |\phi\rangle), \tag{1.13}$$

so we have failed to copy $|\psi\rangle + |\phi\rangle$. In general, if we pick an orthonormal basis, we can copy the basis states, but we will not have correctly copied superpositions of those basis states. We will instead have either measured the original system and therefore destroyed the superposition, or we will have produced a state that is entangled between the original and the "copy." This means that to perform quantum error correction, we cannot simply make backup copies of the quantum state to be preserved. Instead, we must protect the original from any likely error.

1.3 Introduction to Classical Coding Theory

Classical coding theory tends to concentrate on *linear codes*, a subclass of all possible codes with a particular relation between codewords. Suppose we wish to encode k bits using n bits. The data can be represented as a k-dimensional binary vector v. Because we are dealing with binary vectors, all the arithmetic is mod two. For a linear code, the encoded data is then Gv for some $n \times k$ matrix G (with entries from $\mathbf{Z}_2$), which is independent of v. G is called the *generator matrix* for the code. Its columns form a basis for the k-dimensional coding subspace of the n-dimensional binary vector space, and represent basis codewords. The most general possible codeword is an arbitrary linear combination of the basis codewords; thus the name "linear code."

Given a generator matrix G, we can calculate the dual matrix P, which is an $(n-k) \times n$ matrix of 0s and 1s of maximal rank $n-k$ with $PG = 0$. Since any codeword s has the form Gv, $Ps = PGv = 0v = 0$, and P annihilates any codeword. Conversely, suppose $Ps = 0$. Since P has rank $n-k$, it only annihilates a k-dimensional space spanned by the columns of G, and s must be a linear combination of these columns. Thus, $s = Gv$ for some v, and s is a valid codeword. The matrix P is called the *parity check matrix* for the code. It can be used to test if a given vector is a valid codeword, since $Ps = 0$ iff s is a codeword. The *dual code* is defined to be the code with generator matrix P^T and parity matrix G^T.

In order to consider the error-correcting properties of a code, it is useful to look at the *Hamming distance* between codewords. The Hamming distance between two vectors is the minimum number of bits that must be flipped to convert one vector to the other. The distance between a and b is equal to the *weight* (the number of 1s in the vector) of $a+b$. For a code to correct t single-bit errors, it must have distance at least $2t+1$ between any two codewords. A t bit error will take a codeword exactly distance t away from its original value, so when the distance between codewords is at least $2t+1$, we can distinguish errors on different codewords and correct them to the proper codewords. A code to encode k bits in n bits with minimum distance d is said to be an $[n, k, d]$ code.

Now suppose we consider a t bit error. We can write down a vector e to describe this vector by putting ones in the places where bits are flipped and zeros elsewhere. Then if the original codeword is s, after the error it is $s' = s + e$. If we apply the parity check matrix, we get

$$Ps' = P(s+e) = Ps + Pe = 0 + Pe = Pe, \tag{1.14}$$

so the value of Ps' does not depend on the value of s, only on e. If Pe is different for all possible errors e, we will be able to determine precisely what error occurred and fix it. Pe is called the *error syndrome*, since it tells us what the error is. Since $Pe = Pf$ iff $P(e-f) = 0$, to have a code of distance d, we need $Pe \neq 0$ for all vectors e of weight $d-1$ or less. Equivalently, any $d-1$ columns of P must be linearly independent.

We can place upper and lower bounds on the existence of linear codes to correct t errors. Each of the 2^k codewords has a *Hamming sphere* of radius t.

All the words inside the Hamming sphere come from errors acting on the same codeword. For a code on n bits, there are n one-bit errors, $\binom{n}{2}$ two-bit errors, and in general $\binom{n}{j}$ j-bit errors. The Hamming spheres cannot overlap, but they must all fit inside the vector space, which only has 2^n elements. Thus,

$$\sum_{j=0}^{t} \binom{n}{j} 2^k \leq 2^n. \tag{1.15}$$

This is called the *Hamming bound* on $[n, k, 2t+1]$ codes. As n, k, and t get large, this bound approaches the asymptotic form

$$\frac{k}{n} \leq 1 - H\left(\frac{t}{n}\right), \tag{1.16}$$

where $H(x)$ is the *Hamming entropy*

$$H(x) = -x \log_2 x - (1-x)\log_2(1-x). \tag{1.17}$$

We can set a lower bound on the existence of $[n, k, 2t+1]$ linear codes as well, called the *Gilbert-Varshamov* bound. Suppose we have such a code (if necessary with $k = 0$) with

$$\sum_{j=0}^{2t} \binom{n}{j} 2^k < 2^n. \tag{1.18}$$

Then the spheres of distance $2t$ around each codeword do not fill the space, so there is some vector v that is at least distance $2t+1$ from each of the other codewords. In addition, $v + s$ (for any codeword s) is at least distance $2t+1$ from any other codeword s', since the distance is just $(v+s)+s' = v+(s+s')$, which is the distance between v and the codeword $s+s'$. This means that we can add v and all the vectors $v+s$ to the code without dropping the distance below $2t+1$. This gives us an $[n, k+1, 2t+1]$ code. We can continue this process until

$$\sum_{j=0}^{2t} \binom{n}{j} 2^k \geq 2^n. \tag{1.19}$$

Asymptotically, this becomes

$$\frac{k}{n} \geq 1 - H\left(\frac{2t}{n}\right). \tag{1.20}$$

Another case of great interest is the capacity of a classical channel. This is equal to the *efficiency* k/n of the most efficient code on an asymptotically large block that corrects measure one of the errors occuring. For instance, a common channel is the *binary symmetric channel*, where an error occurs independently on each bit with probability p for both 0 and 1. Shannon showed that channel capacity is just equal to one minus the entropy introduced by the channel [15]. For the binary symmetric channel, the entropy is just the Hamming entropy

$H(p)$, so the capacity is $1 - H(p)$, coinciding with the Hamming bound for the expected number of errors $t = pn$. Shannon also showed that the capacity of a channel can be achieved by choosing codewords at random, then discarding only a few of them (measure zero asymptotically).

Chapter 2

Basics of Quantum Error Correction

2.1 The Quantum Channel

Now we turn to the quantum channel. A noisy quantum channel can be a regular communications channel which we expect to preserve at least some degree of quantum coherence, or it can be the passage of time as a set of qubits sits around, interacting with its environment, or it can be the result of operating with a noisy gate on some qubits in a quantum computer. In any of these cases, the input of a pure quantum state can produce a mixed state as output as the data qubits become entangled with the environment. Even when a pure state comes out, it might not be the same state as the one that went in.

At first it appears that trying to correct a mixed state back into the correct pure state is going to be harder than correcting an erroneous pure state, but this is not the case. The output mixed state can be considered as an ensemble of pure states. If we can correct each of the pure states in the ensemble back to the original input state, we have corrected the full mixed state. Another way of phrasing this is to say the channel applies a superoperator to the input density matrix. We can diagonalize this superoperator and write it as the direct sum of a number of different matrices acting directly on the possible input pure states with various probabilities. If the code can correct any of the possible matrices, it can correct the full superoperator. A key point is that the individual matrices need not be unitary. From now on, I will only consider the effects of a (possibly non-unitary) matrix acting on a pure state.

2.2 A Simple Code

For the moment, let us consider only channels which cause an error on a single qubit at a time. We wish to protect a single logical qubit against error. We

cannot send it through the channel as is, because the one qubit that is affected might be the one we want to keep. Suppose we send through nine qubits after encoding the logical qubit as follows:

$$|0\rangle \rightarrow |\overline{0}\rangle = (|000\rangle + |111\rangle)(|000\rangle + |111\rangle)(|000\rangle + |111\rangle) \quad (2.1)$$
$$|1\rangle \rightarrow |\overline{1}\rangle = (|000\rangle - |111\rangle)(|000\rangle - |111\rangle)(|000\rangle - |111\rangle). \quad (2.2)$$

The data is no longer stored in a single qubit, but instead spread out among nine of them. Note that even if we know the nine qubits are in one of these two states, we cannot determine which one without making a measurement on at least three qubits. This code is due to Shor [10].

Suppose the channel flips a single qubit, say the first one, switching $|0\rangle$ and $|1\rangle$. Then by comparing the first two qubits, we find they are different, which is not allowed for any valid codeword. Therefore we know an error occurred, and furthermore, it flipped either the first or second qubit. Note that we do not actually measure the first and second qubits, since this would destroy the superposition in the codeword; we just measure the difference between them.

Now we compare the first and third qubits. Since the first qubit was flipped, it will disagree with the third; if the second qubit had been flipped, the first and third would have agreed. Therefore, we have narrowed down the error to the first qubit and we can fix it simply by flipping it back. To handle possible bit flips on the other blocks of three, we do the same comparisons inside the other blocks.

However, this is not the only sort of error that could have occurred. The channel might have left the identity of the 0 and 1 alone, but altered their relative phase, introducing, for instance, a relative factor of -1 when the first qubit is $|1\rangle$. Then the two basis states become

$$|\overline{0}\rangle \rightarrow (|000\rangle - |111\rangle)(|000\rangle + |111\rangle)(|000\rangle + |111\rangle) \quad (2.3)$$
$$|\overline{1}\rangle \rightarrow (|000\rangle + |111\rangle)(|000\rangle - |111\rangle)(|000\rangle - |111\rangle). \quad (2.4)$$

By comparing the sign of the first block of three with the second block of three, we can see that a sign error has occurred in one of those blocks. Then by comparing the signs of the first and third blocks of three, we narrow the sign error down to the first block, and flip the sign back to what it should be. Again, we do not want to actually measure the signs, only whether they agree. In this case, measuring the signs would give us information about whether the state is $|\overline{0}\rangle$ or $|\overline{1}\rangle$, which would destroy any superposition between them.

This does not exhaust the list of possible one qubit errors. For instance, we could have both a bit flip and a sign flip on the same qubit. However, by going through both processes described above, we will fix first the bit flip, then the sign flip (in fact, this code will correct a bit flip and a sign flip even if they are on different qubits). The original two errors can be described as the operation of

$$\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \text{ and } \sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \quad (2.5)$$

The simultaneous bit and sign flip is

$$\sigma_y = i\sigma_x\sigma_z = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}. \tag{2.6}$$

Sometimes I will write σ_{xi}, σ_{yi}, or σ_{zi} to represent σ_x, σ_y, or σ_z acting on the ith qubit.

The most general one-qubit error that can occur is some 2×2 matrix; but such a matrix can always be written as the (complex) linear combination of σ_x, σ_y, σ_z, and the 2×2 identity matrix I. Consider what happens to the code when such an error occurs:

$$|\psi\rangle = \alpha|\overline{0}\rangle + \beta|\overline{1}\rangle \rightarrow a\sigma_{xi}|\psi\rangle + b\sigma_{yi}|\psi\rangle + c\sigma_{zi}|\psi\rangle + d|\psi\rangle. \tag{2.7}$$

Suppose we perform the process above, comparing bits within a block of three, and comparing the signs of blocks of three. This acts as a measurement of which error (or the identity) has occurred, causing the state, originally in a superposition, to collapse to $\sigma_{xi}|\psi\rangle$ with probability $|a|^2$, to $\sigma_{yi}|\psi\rangle$ with probability $|b|^2$, to $\sigma_{zi}|\psi\rangle$ with probability $|c|^2$, and to $|\psi\rangle$ with probability $|d|^2$. In any of the four cases, we have determined which error occurred and we can fix it.

2.3 Properties of Any Quantum Code

Now let us consider properties of more general codes. A code to encode k qubits in n qubits will have 2^k basis codewords corresponding to the basis of the original states. Any linear combination of these basis codewords is also a valid codeword, corresponding to the same linear combination of the unencoded basis states. The space T of valid codewords (the *coding space*) is therefore a Hilbert space in its own right, a subspace of the full 2^n-dimensional Hilbert space. As with Shor's nine-qubit code, if we can correct errors E and F, we can correct $aE + bF$, so we only need to consider whether the code can correct a basis of errors. One convenient basis to use is the set of tensor products of σ_x, σ_y, σ_z, and I. The *weight* of an operator of this form is the number of qubits on which it differs from the identity. The set of all these tensor products with a possible overall factor of -1 or $\pm i$ forms a group $\mathcal{G}$ under multiplication. $\mathcal{G}$ will play a major role in the stabilizer formalism. Sometimes I will write it $\mathcal{G}_n$ to distinguish the groups for different numbers of qubits. $\mathcal{G}_1$ is just the quaternionic group; $\mathcal{G}_n$ is the direct product of n copies of the quaternions modulo all but a global phase factor.

In order for the code to correct two errors E_a and E_b, we must always be able to distinguish error E_a acting on one basis codeword $|\psi_i\rangle$ from error E_b acting on a different basis codeword $|\psi_j\rangle$. We can only be sure of doing this if $E_a|\psi_1\rangle$ is orthogonal to $E_b|\psi_2\rangle$; otherwise there is some chance of confusing them. Thus,

$$\langle\psi_i|E_a^\dagger E_b|\psi_j\rangle = 0 \tag{2.8}$$

when $i \neq j$ for correctable errors E_a and E_b. Note that we normally include the identity in the set of possible "errors," since we do not want to confuse an error

on one qubit with nothing happening to another. If we have a channel in which we are certain *some* error occurred, we do not need to include the identity as a possible error. In any case, the set of correctable errors is unlikely to be a group — it does not even need to be closed under multiplication.

However, (2.8) is insufficient to guarantee a code will work as a quantum error-correcting code. When we make a measurement to find out about the error, we must learn nothing about the actual state of the code within the coding space. If we did learn something, we would be disturbing superpositions of the basis states, so while we might correct the basis states, we would not be correcting an arbitrary valid codeword. We learn information about the error by measuring $\langle\psi_i|E_a^\dagger E_b|\psi_i\rangle$ for all possible errors E_a and E_b. This quantity must therefore be the same for all the basis codewords:

$$\langle\psi_i|E_a^\dagger E_b|\psi_i\rangle = \langle\psi_j|E_a^\dagger E_b|\psi_j\rangle. \tag{2.9}$$

We can combine equations (2.8) and (2.9) into a single equation:

$$\langle\psi_i|E_a^\dagger E_b|\psi_j\rangle = C_{ab}\delta_{ij}, \tag{2.10}$$

where $|\psi_i\rangle$ and $|\psi_j\rangle$ run over all possible basis codewords, E_a and E_b run over all possible errors, and C_{ab} is independent of i and j. This condition was found by Knill and Laflamme [16] and Bennett *et al.* [17].

The above argument shows that (2.10) is a necessary condition for the code to correct the errors $\{E_a\}$. It is also a sufficient condition: The matrix C_{ab} is Hermitian, so it can be diagonalized. If we do this and rescale the errors $\{E_a\}$ appropriately, we get a new basis $\{F_a\}$ for the space of possible errors, with either

$$\langle\psi_i|F_a^\dagger F_b|\psi_j\rangle = \delta_{ab}\delta_{ij} \tag{2.11}$$

or

$$\langle\psi_i|F_a^\dagger F_b|\psi_j\rangle = 0, \tag{2.12}$$

depending on a. Note that this basis will not necessarily contain operators that are tensor products of one-qubit operators. Errors of the second type actually annihilate any codeword, so the probability of one occuring is strictly zero and we need not consider them. The other errors always produce orthogonal states, so we can make some measurement that will tell us exactly which error occurred, at which point it is a simple matter to correct it. Therefore, a code satisfies equation (2.10) for all E_a and E_b in some set $\mathcal{E}$ iff the code can correct all errors in $\mathcal{E}$.

Another minor basis change allows us to find a basis where any two errors acting on a given codeword either produce orthogonal states or exactly the same state. The errors F_a that annihilate codewords correspond to two errors that act the same way on codewords. For instance, in Shor's nine-qubit code, σ_{z1} and σ_{z2} act the same way on the code, so $\sigma_{z1} - \sigma_{z2}$ will annihilate codewords. This phenomenon will occur iff C_{ab} does not have maximum rank. A code for which C_{ab} is singular is called a *degenerate* code, while a code for which it is not is *nondegenerate*. Shor's nine-qubit code is degenerate; we will see many

examples of nondegenerate codes later. Note that whether a code is degenerate or not depends on the set of errors it is intended to correct. For instance, a two-error-correcting degenerate code might be nondegenerate when considered as a one-error-correcting code.

In equation (2.10), $E = E_a^\dagger E_b$ is still in the group $\mathcal{G}$ when E_a and E_b are in $\mathcal{G}$. The weight of the smallest E in $\mathcal{G}$ for which (2.10) does *not* hold is called the *distance* of the code. A quantum code to correct up to t errors must have distance at least $2t + 1$. Every code has distance at least one. A distance d code encoding k qubits in n qubits is described as an $[n, k, d]$ code. Note that a quantum $[n, k, d]$ code is often written in the literature as $[[n, k, d]]$ to distinguish it from a classical $[n, k, d]$ code. I have chosen the notation $[n, k, d]$ to emphasize the similarities with the classical theory; when I need to distinguish, I will do so using the words "quantum" and "classical."

We can also consider variations of the usual error-correction problem. For instance, suppose we only want to detect if an error has occurred, not to correct it. This could, for instance, be used to prevent errors using the quantum Zeno effect [18]. In this case, we do not need to distinguish error E_a from E_b, only from the identity. We can use the same argument to find (2.10), only now $E_b = I$ always. This means a code to detect s errors must have distance at least $s + 1$. Another variation is when we know in which qubit(s) an error has occurred, as in the quantum erasure channel [19]. In this case, we only need distinguish E_a from those E_b affecting the same qubits. This means that $E_a^\dagger E_b$ has the same weight as E_a, and to correct r such located errors, we need a code of distance at least $r + 1$. We can also imagine combining all of these tasks. A code to correct t arbitrary errors, r additional located errors, and detect a further s errors must have distance at least $r + s + 2t + 1$.

2.4 Error Models

In this thesis, I will mostly assume that errors occur independently on different qubits, and that when an error occurs on a qubit, it is equally likely to be a σ_x, σ_y, or σ_z error. If the probability ϵ of error per qubit is fairly small, it is often useful to simply ignore the possibility of more than t errors, since this only occurs with probability $O(\epsilon^{t+1})$. Thus, I will typically deal with codes that correct up to t arbitrary errors. Such a code will handle any error on up to t qubits that leaves the data somewhere in the normal computational space (although moving it outside of the space of valid codewords).

In some systems, there will be errors that move the system outside of the computational space. For instance, if the data is stored as the ground or metastable excited state of an ion, the electron might instead end up in a different excited state. If the data is stored in the polarization of a photon, the photon might escape. In both of these cases, the normal error correction networks will not function properly, since they assume that the qubit is either in the state $|0\rangle$ or $|1\rangle$. However, by performing some measurement that distinguishes between the computational Hilbert space and other possible states, we can determine

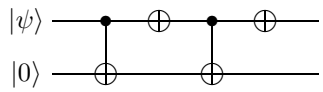

Figure 2.1: Network to detect leakage errors.

not only that this sort of *leakage error* has occurred, but also on which qubit it has occurred. Then we can cool the atom to the ground state or introduce a new photon with random polarization, and the error becomes a located error, which was discussed at the end of the previous section. One possible network of gates to detect a leakage error is given in figure 2.1 (see appendix A for a description of the symbols used in this and later figures). This network asssumes that states outside the normal computational space do not interact at all with other qubits. If the data state $|\psi\rangle$ is either $|0\rangle$ or $|1\rangle$, the ancilla qubit will flip and become $|1\rangle$. If the data state is neither $|0\rangle$ nor $|1\rangle$, the ancilla will remain $|0\rangle$, thus signalling a leakage error on this data qubit.

Another possible difficulty arises when correlated errors on multiple qubits can occur. While this can in principle be a severe problem, it can be handled without a change in formalism as long as the chance of a correlated error drops rapidly enough with the size of the blocks of errors. Since a t-qubit error will occur with probability $O(\epsilon^t)$ when the probability of uncorrelated single-qubit errors is ϵ, as long as the probability of a t-qubit correlated error is $O(\epsilon^t)$, the correlated errors cause no additional problems.

In real systems, the assumption that errors are equally likely to be σ_x, σ_y, and σ_z errors is a poor one. In practice, some linear combinations of σ_x, σ_y, and σ_z are going to be more likely than others. For instance, when the qubits are ground or excited states of an ion, a likely source of errors is spontaneous emission. After some amount of time, the excited state will either decay to the ground state, producing the error $\sigma_x + i\sigma_y$ with probability ϵ, or it will not, which changes the relative amplitudes of $|0\rangle$ and $|1\rangle$, resulting in the error $I - \sigma_z$ with probability $O(\epsilon^2)$. A channel that performs this sort of time evolution is known as an *amplitude damping* channel. Since the only $O(1)$ effect of time evolution is the identity, this sort of error can be protected against to lowest order by a code to correct an arbitrary single error. However, codes that take account of the restricted possibilities for errors can be more efficient than codes that must correct a general error [20], and understanding the physically likely sources of error will certainly be an important part of engineering quantum computers.

Chapter 3

Stabilizer Coding

3.1 The Nine-Qubit Code Revisited

Let us look more closely at the procedure we used to correct errors for the nine-qubit code. To detect a bit flip error on one of the first three qubits, we compared the first two qubits and the first and third qubits. This is equivalent to measuring the eigenvalues of $\sigma_{z1}\sigma_{z2}$ and $\sigma_{z1}\sigma_{z3}$. If the first two qubits are the same, the eigenvalue of $\sigma_{z1}\sigma_{z2}$ is $+1$; if they are different, the eigenvalue is -1. Similarly, to detect a sign error, we compare the signs of the first and second blocks of three and the first and third blocks of three. This is equivalent to measuring the eigenvalues of $\sigma_{x1}\sigma_{x2}\sigma_{x3}\sigma_{x4}\sigma_{x5}\sigma_{x6}$ and $\sigma_{x1}\sigma_{x2}\sigma_{x3}\sigma_{x7}\sigma_{x8}\sigma_{x9}$. Again, if the signs agree, the eigenvalues will be $+1$; if they disagree, the eigenvalues will be -1. In order to totally correct the code, we must measure the eigenvalues of a total of eight operators. They are listed in table 3.1.

The two valid codewords $|\overline{0}\rangle$ and $|\overline{1}\rangle$ in Shor's code are eigenvectors of all eight of these operators with eigenvalue $+1$. All the operators in $\mathcal{G}$ that fix both $|\overline{0}\rangle$ and $|\overline{1}\rangle$ can be written as the product of these eight operators. The set of operators that fix $|\overline{0}\rangle$ and $|\overline{1}\rangle$ form a group S, called the *stabilizer* of the code, and M_1 through M_8 are the generators of this group.

When we measure the eigenvalue of M_1, we determine if a bit flip error has

M_1	σ_z	σ_z	I	I	I	I	I	I	I
M_2	σ_z	I	σ_z	I	I	I	I	I	I
M_3	I	I	I	σ_z	σ_z	I	I	I	I
M_4	I	I	I	σ_z	I	σ_z	I	I	I
M_5	I	I	I	I	I	I	σ_z	σ_z	I
M_6	I	I	I	I	I	I	σ_z	I	σ_z
M_7	σ_x	σ_x	σ_x	σ_x	σ_x	σ_x	I	I	I
M_8	σ_x	σ_x	σ_x	I	I	I	σ_x	σ_x	σ_x

Table 3.1: The stabilizer for Shor's nine-qubit code

occurred on qubit one or two, i.e., if σ_{x1} or σ_{x2} has occurred. Note that both of these errors anticommute with M_1, while σ_{x3} through σ_{x9}, which cannot be detected by just M_1, commute with it. Similarly, M_2 detects σ_{x1} or σ_{x3}, which anticommute with it, and M_7 detects σ_{z1} through σ_{z6}. In general, if $M \in S$, $\{M, E\} = 0$, and $|\psi\rangle \in T$, then

$$ME|\psi\rangle = -EM|\psi\rangle = -E|\psi\rangle, \tag{3.1}$$

so $E|\psi\rangle$ is an eigenvector of M with eigenvalue -1 instead of $+1$ and to detect E we need only measure M.

The distance of this code is in fact three. Even a cursory perusal reveals that any single-qubit operator σ_{xi}, σ_{yi}, or σ_{zi} will anticommute with one or more of M_1 through M_8. Since states with different eigenvalues are orthogonal, condition (2.10) is satisfied when E_a has weight one and $E_b = I$. We can also check that every two-qubit operator E anticommutes with some element of S, except for those of the form $\sigma_{za}\sigma_{zb}$ where a and b are in the same block of three. However, the operators of this form are actually in the stabilizer. This means that $\sigma_{za}\sigma_{zb}|\psi\rangle = |\psi\rangle$ for any codeword $|\psi\rangle$, and $\langle\psi|\sigma_{za}\sigma_{zb}|\psi\rangle = \langle\psi|\psi\rangle = 1$ for all codewords $|\psi\rangle$, and these operators also satisfy equation (2.10). Since $\sigma_{za}\sigma_{zb}$ is in the stabilizer, both σ_{za} and σ_{zb} act the same way on the codewords, and there is no need to distinguish them. When we get to operators of weight three, we do find some for which (2.10) fails. For instance, $\sigma_{x1}\sigma_{x2}\sigma_{x3}$ commutes with everything in S, but

$$\langle\overline{0}|\sigma_{x1}\sigma_{x2}\sigma_{x3}|\overline{0}\rangle = +1 \tag{3.2}$$

$$\langle\overline{1}|\sigma_{x1}\sigma_{x2}\sigma_{x3}|\overline{1}\rangle = -1. \tag{3.3}$$

3.2 The General Stabilizer Code

The stabilizer construction applies to many more codes than just the nine-qubit one [21, 22]. In general, the stabilizer S is some Abelian subgroup of $\mathcal{G}$ and the coding space T is the space of vectors fixed by S. Since σ_y has imaginary components, while σ_x and σ_z are real, with an even number of σ_y's in each element of the stabilizer, all the coefficients in the basis codewords can be chosen to be real; if there are an odd number of σ_y's, they may be imaginary. However, Rains has shown that whenever a (possibly complex) code exists, a real code exists with the same parameters [23]. Therefore, I will largely restrict my attention to real codes.

For a code to encode k qubits in n, T has 2^k dimensions and S has 2^{n-k} elements. S must be an Abelian group, since only commuting operators can have simultaneous eigenvectors, but provided it is Abelian and neither i nor -1 is in S, the space $T = \{|\psi\rangle \text{ s.t. } M|\psi\rangle = |\psi\rangle \ \forall M \in S\}$ does have dimension 2^k. At this point it will be helpful to note a few properties of $\mathcal{G}$. Since $\sigma_x^2 = \sigma_y^2 = \sigma_z^2 = +1$, every element in $\mathcal{G}$ squares to ± 1. Also, σ_x, σ_y, and σ_z on the same qubit anticommute, while they commute on different qubits. Therefore, any two elements of $\mathcal{G}$ either commute or they anticommute. σ_x, σ_y, and σ_z are all

Hermitian, but of course $(iI)^\dagger = -iI$, so elements of $\mathcal{G}$ can be either Hermitian or anti-Hermitian. In either case, if $A \in \mathcal{G}$, $A^\dagger \in G$ also. Similarly, σ_x, σ_y, and σ_z are all unitary, so every element of $\mathcal{G}$ is unitary.

As before, if $M \in S$, $|\psi_i\rangle \in T$, and $\{M, E\} = 0$, then $ME|\psi_i\rangle = -E|\psi_i\rangle$, so

$$\langle\psi_i|E|\psi_j\rangle = \langle\psi_i|ME|\psi_j\rangle = -\langle\psi_i|E|\psi_j\rangle = 0. \tag{3.4}$$

Therefore the code satisfies (2.8) whenever $E = E_a^\dagger E_b = \pm E_a E_b$ anticommutes with M for some $M \in S$. In fact, in such a case it also satisfies (2.9), since $\langle\psi_i|E|\psi_i\rangle = \langle\psi_j|E|\psi_j\rangle = 0$. Therefore, if $E_a^\dagger E_b$ anticommutes with some element of S for all errors E_a and E_b in some set, the code will correct that set of errors.

Of course, strictly speaking, this is unlikely to occur. Generally, I will be an allowed error, and $E = I^\dagger I$ commutes with everything. However, S is a group, so $I \in S$. In general, if $E \in S$,

$$\langle\psi_i|E|\psi_j\rangle = \langle\psi_i|\psi_j\rangle = \delta_{ij}. \tag{3.5}$$

This will satisfy equation (2.10) also.

Now, there generally are many elements of $\mathcal{G}$ that commute with everything in S but are not actually in S. The set of elements in $\mathcal{G}$ that commute with all of S is defined as the centralizer $C(S)$ of S in $\mathcal{G}$. Because of the properties of S and $\mathcal{G}$, the centralizer is actually equal to the normalizer $N(S)$ of S in $\mathcal{G}$, which is defined as the set of elements of $\mathcal{G}$ that fix S under conjugation. To see this, note that for any $A \in \mathcal{G}$, $M \in S$,

$$A^\dagger M A = \pm A^\dagger A M = \pm M. \tag{3.6}$$

Since $-1 \notin S$, $A \in N(S)$ iff $A \in C(S)$, so $N(S) = C(S)$. Note that $S \subseteq N(S)$. In fact, S is a normal subgroup of $N(S)$. $N(S)$ contains $4 \cdot 2^{n+k}$ elements. The factor of four is for the overall phase factor. Since an overall phase has no effect on the physical quantum state, often, when considering $N(S)$, I will only really consider $N(S)$ without this global phase factor.

If $E \in N(S) - S$, then E rearranges elements of T but does not take them out of T: if $M \in S$ and $|\psi\rangle \in T$, then

$$ME|\psi\rangle = EM|\psi\rangle = E|\psi\rangle, \tag{3.7}$$

so $E|\psi\rangle \in T$ also. Since $E \notin S$, there is some state in T that is not fixed by E. Unless it differs from an element of S by an overall phase, E will therefore be undetectable by this code.

Putting these considerations together, we can say that a quantum code with stabilizer S will detect all errors E that are either in S or anticommute with some element of S. In other words, $E \in S \cup (\mathcal{G} - N(S))$. This code will correct any set of errors $\{E_i\}$ iff $E_a E_b \in S \cup (\mathcal{G} - N(S))$ $\forall E_a, E_b$ (note that $E_a^\dagger E_b$ commutes with $M \in \mathcal{G}$ iff $E_a E_b = \pm E_a^\dagger E_b$ does). For instance, the code will have distance d iff $N(S) - S$ contains no elements of weight less than d. If S has elements of weight less than d (except the identity), it is a degenerate

code; otherwise it is a nondegenerate code. For instance, the nine-qubit code is degenerate, since it has distance three and $\sigma_{z1}\sigma_{z2} \in S$. A nondegenerate stabilizer code satisfies

$$\langle\psi_i|E_a^\dagger E_b|\psi_j\rangle = \delta_{ab}\delta_{ij}. \tag{3.8}$$

By convention, an $[n, 0, d]$ code must be nondegenerate. When $E_aE_b \in S$, we say that the errors E_a and E_b are degenerate. We cannot distinguish between E_a and E_b, but there is no need to, since they have the same effect on the codewords.

It is sometimes useful to define the *error syndrome* for a stabilizer code. Let $f_M : \mathcal{G} \to \mathbf{Z}_2$,

$$f_M(E) = \begin{cases} 0 & \text{if } [M, E] = 0 \\ 1 & \text{if } \{M, E\} = 0 \end{cases} \tag{3.9}$$

and $f(E) = (f_{M_1}(E), \ldots, f_{M_{n-k}}(E))$, where $M_1, \ldots, M_{n-k}$ are the generators of S. Then $f(E)$ is some $(n-k)$-bit binary number which is 0 iff $E \in N(S)$. $f(E_a) = f(E_b)$ iff $f(E_aE_b) = 0$, so for a nondegenerate code, $f(E)$ is different for each correctable error E.

In order to perform the error-correction operation for a stabilizer code, all we need to do is measure the eigenvalue of each generator of the stabilizer. The eigenvalue of M_i will be $(-1)^{f_{M_i}(E)}$, so this process will give us the error syndrome. The error syndrome in turn tells us exactly what error occurred (for a nondegenerate code) or what set of degenerate errors occurred (for a degenerate code). The error will always be in $\mathcal{G}$ since the code uses that error basis, and every operator in $\mathcal{G}$ is unitary, and therefore invertible. Then we just apply the error operator (or one equivalent to it by multiplication by S) to fix the state. Note that even if the original error that occurred is a nontrivial linear combination of errors in $\mathcal{G}$, the process of syndrome measurement will project onto one of the basis errors. If the resulting error is not in the correctable set, we will end up in the wrong encoded state, but otherwise, we are in the correct state. In chapter 5, I describe a few ways of measuring the error syndrome that are tolerant of imperfect component gates.

Since the elements of $N(S)$ move codewords around within T, they have a natural interpretation as encoded operations on the codewords. Since S fixes T, actually only $N(S)/S$ will act on T nontrivially. If we pick a basis for T consisting of eigenvectors of n commuting elements of $N(S)$, we get an automorphism $N(S)/S \to \mathcal{G}_k$. $N(S)/S$ can therefore be generated by i (which we will by and large ignore) and $2k$ equivalence classes, which I will write $\overline{X}_i$ and $\overline{Z}_i$ $(i = 1 \ldots k)$, where $\overline{X}_i$ maps to σ_{xi} in $\mathcal{G}_k$ and $\overline{Z}_i$ maps to σ_{zi} in $\mathcal{G}_k$. They are encoded σ_x and σ_z operators for the code. If $k = 1$, I will write $\overline{X}_1 = \overline{X}$ and $\overline{Z}_1 = \overline{Z}$. The $\overline{X}$ and $\overline{Z}$ operators satisfy

$$[\overline{X}_i, \overline{X}_j] = 0 \tag{3.10}$$

$$[\overline{Z}_i, \overline{Z}_j] = 0 \tag{3.11}$$

$$[\overline{X}_i, \overline{Z}_j] = 0 \ (i \neq j) \tag{3.12}$$

$$\{\overline{X}_i, \overline{Z}_i\} = 0. \tag{3.13}$$

M_1	σ_x	σ_z	σ_z	σ_x	I
M_2	I	σ_x	σ_z	σ_z	σ_x
M_3	σ_x	I	σ_x	σ_z	σ_z
M_4	σ_z	σ_x	I	σ_x	σ_z
$\overline{X}$	σ_x	σ_x	σ_x	σ_x	σ_x
$\overline{Z}$	σ_z	σ_z	σ_z	σ_z	σ_z

Table 3.2: The stabilizer for the five-qubit code.

3.3 Some Examples

I shall now present a few short codes to use as examples. The first encodes one qubit in five qubits [17, 24] and is given in table 3.2. I have also included $\overline{X}$ and $\overline{Z}$, which, along with M_1 through M_4, generate $N(S)$. Note that this code is *cyclic* (i.e., the stabilizer and codewords are invariant under cyclic permutations of the qubits). It has distance three (for instance, $\sigma_{y1}\sigma_{z2}\sigma_{y3} \in N(S) - S$) and is nondegenerate. We can take the basis codewords for this code to be

$$|\overline{0}\rangle = \sum_{M \in S} M \ |00000\rangle \tag{3.14}$$

and

$$|\overline{1}\rangle = \overline{X}|\overline{0}\rangle. \tag{3.15}$$

That is,

$$\begin{aligned} |\overline{0}\rangle \;=\; & |00000\rangle + M_1|00000\rangle + M_2|00000\rangle + M_3|00000\rangle + M_4|00000\rangle \\ & + M_1M_2|00000\rangle + M_1M_3|00000\rangle + M_1M_4|00000\rangle \\ & + M_2M_3|00000\rangle + M_2M_4|00000\rangle + M_3M_4|00000\rangle \qquad (3.16) \\ & + M_1M_2M_3|00000\rangle + M_1M_2M_4|00000\rangle + M_1M_3M_4|00000\rangle \\ & + M_2M_3M_4|00000\rangle + M_1M_2M_3M_4|00000\rangle \\ = \; & |00000\rangle + |10010\rangle + |01001\rangle + |10100\rangle \\ & + |01010\rangle - |11011\rangle - |00110\rangle - |11000\rangle \\ & - |11101\rangle - |00011\rangle - |11110\rangle - |01111\rangle \qquad (3.17) \\ & - |10001\rangle - |01100\rangle - |10111\rangle + |00101\rangle, \end{aligned}$$

and

$$\begin{aligned} |\overline{1}\rangle \;=\; & \overline{X}|\overline{0}\rangle \\ = \; & |11111\rangle + |01101\rangle + |10110\rangle + |01011\rangle \\ & + |10101\rangle - |00100\rangle - |11001\rangle - |00111\rangle \\ & - |00010\rangle - |11100\rangle - |00001\rangle - |10000\rangle \qquad (3.18) \\ & - |01110\rangle - |10011\rangle - |01000\rangle + |11010\rangle. \end{aligned}$$

Since multiplying by an element of the stabilizer merely rearranges the sum $\sum M$, these two states are in T. When these are the encoded 0 and 1, $\overline{X}$ is the

M_1	σ_x	σ_x	σ_x	σ_x	σ_x	σ_x	σ_x	σ_x
M_2	σ_z	σ_z	σ_z	σ_z	σ_z	σ_z	σ_z	σ_z
M_3	I	σ_x	I	σ_x	σ_y	σ_z	σ_y	σ_z
M_4	I	σ_x	σ_z	σ_y	I	σ_x	σ_z	σ_y
M_5	I	σ_y	σ_x	σ_z	σ_x	σ_z	I	σ_y
$\overline{X}_1$	σ_x	σ_x	I	I	I	σ_z	I	σ_z
$\overline{X}_2$	σ_x	I	σ_x	σ_z	I	I	σ_z	I
$\overline{X}_3$	σ_x	I	I	σ_z	σ_x	σ_z	I	I
$\overline{Z}_1$	I	σ_z	I	σ_z	I	σ_z	I	σ_z
$\overline{Z}_2$	I	I	σ_z	σ_z	I	I	σ_z	σ_z
$\overline{Z}_3$	I	I	I	I	σ_z	σ_z	σ_z	σ_z

Table 3.3: The stabilizer for the eight-qubit code.

encoded bit flip operator σ_x and $\overline{Z}$ is the encoded σ_z. This code also has the property that every possible error syndrome is used by the single-qubit errors. It is therefore a *perfect* code. There are a number of other perfect codes [25, 26], which will be discussed in chapter 8.

A code encoding three qubits in eight qubits [21, 22, 27] appears in table 3.3. Again, M_1 through M_5 generate the stabilizer, and generate $N(S)$ with $\overline{X}_i$ and $\overline{Z}_i$. This is also a nondegenerate distance three code. The codewords are

$$|\overline{c_1 c_2 c_3}\rangle = \overline{X}_1^{c_1}\overline{X}_2^{c_2}\overline{X}_3^{c_3}\sum_{M\in S} M|00000000\rangle. \tag{3.19}$$

The operators $\overline{X}_i$ and $\overline{Z}_i$ are the encoded σ_x and σ_z on the ith encoded qubit. This code is one of an infinite family of codes [21, 28], which I present in chapter 8.

A particularly useful class of codes with simple stabilizers is the Calderbank-Shor-Steane (or *CSS*) class of codes [29, 30]. Suppose we have a classical code with parity check matrix P. We can make a quantum code to correct just σ_x errors using a stabilizer with elements corresponding to the rows of P, with a σ_z wherever P has a 1 and I's elsewhere. The error syndrome $f(E)$ for a product of σ_x errors E is then equal to the classical error syndrome for the same set of classical bit flip errors. Now add in stabilizer generators corresponding to the parity check matrix Q of a second classical code, only now with σ_x's instead of σ_z's. These generators will identify σ_z errors. Together, they can also identify σ_y errors, which will have a nontrivial error syndrome for both parts. In general, a code formed this way will correct as many σ_x errors as the code for P can correct, and as many σ_z errors as the code for Q can correct; a σ_y error counts as one of each.

We can only combine P and Q into a single stabilizer in the CSS form if the generators derived from the two codes commute. This will be true iff the rows of P and Q are orthogonal using the binary dot product. This means that the dual code of each code must be a subset of the other code. The minimum distance of the quantum code will be the minimum of the distances of P and

M_1	σ_x	σ_x	σ_x	σ_x	I	I	I
M_2	σ_x	σ_x	I	I	σ_x	σ_x	I
M_3	σ_x	I	σ_x	I	σ_x	I	σ_x
M_4	σ_z	σ_z	σ_z	σ_z	I	I	I
M_5	σ_z	σ_z	I	I	σ_z	σ_z	I
M_6	σ_z	I	σ_z	I	σ_z	I	σ_z
$\overline{X}$	I	I	I	I	σ_x	σ_x	σ_x
$\overline{Z}$	I	I	I	I	σ_z	σ_z	σ_z

Table 3.4: The seven-qubit CSS code.

Q. An example of a code of this sort is given in table 3.4. It is based on the classical $[7, 4, 3]$ Hamming code, which is self-dual. For this code, the codewords are

$$\begin{aligned} |\overline{0}\rangle &= |0000000\rangle + |1111000\rangle + |1100110\rangle + |1010101\rangle \\ &\quad + |0011110\rangle + |0101101\rangle + |0110011\rangle + |1001011\rangle \end{aligned} \tag{3.20}$$

and

$$\begin{aligned} |\overline{1}\rangle &= |0000111\rangle + |1111111\rangle + |1100001\rangle + |1010010\rangle \\ &\quad + |0011001\rangle + |0101010\rangle + |0110100\rangle + |1001100\rangle. \end{aligned} \tag{3.21}$$

The encoded $|0\rangle$ state is the superposition of the even codewords in the Hamming code and the encoded $|1\rangle$ state is the superposition of the odd codewords in the Hamming code. This behavior is characteristic of CSS codes; in general, the various quantum codewords are superpositions of the words in subcodes of one of the classical codes.

CSS codes are not as efficient as the most general quantum code, but they are easy to derive from known classical codes and their simple form often makes them ideal for other purposes. For instance, the seven-qubit code is particularly well suited for fault-tolerant computation (as I will discuss in chapter 5).

3.4 Alternate Languages for Stabilizers

There are number of possible ways of describing the stabilizer of a quantum code. They each have advantages and are useful in different circumstances. The description I have used so far uses the language of finite group theory and is particularly useful for making contact with the usual language of quantum mechanics. This is the form presented in [21].

We can instead write the stabilizer using binary vector spaces, as in [22], which emphasizes connections with the classical theory of error-correcting codes. To do this, we write the stabilizer as a pair of $(n-k) \times n$ binary matrices (or often one $(n-k) \times 2n$ matrix with a line separating the two halves). The rows correspond to the different generators of the stabilizer and the columns

correspond to different qubits. One matrix has a 1 whenever the generator has a σ_x or a σ_y in the appropriate place, the other has a 1 whenever the generator has a σ_y or σ_z. Overall phase factors get dropped. For instance, the five-qubit code in this form becomes

$$\left(\begin{array}{ccccc|ccccc} 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{array}\right). \tag{3.22}$$

Other elements of $\mathcal{G}$ get converted to two n-dimensional vectors in the same way. We can convert back to the group theory formalism by writing down operators with a σ_x if the left vector or matrix has a 1, a σ_z if the right vector or matrix has a 1, and a σ_y if they are both 1. The generators formed this way will never have overall phase factors, although other elements of the group might. Multiplication of group elements corresponds to addition of the corresponding binary vectors.

In the binary formalism, the condition that two operators commute with each other becomes the condition that the following inner product is 0:

$$Q(a|b,c|d) = \sum_{i=1}^{n} (a_i d_i + b_i c_i) = 0, \tag{3.23}$$

using binary arithmetic as usual. a_i, b_i, c_i, and d_i are the ith components of the corresponding vectors. Therefore the condition that the stabilizer be Abelian converts to the condition that the stabilizer matrix $(A|B)$ satisfy

$$\sum_{l=1}^{n} (A_{il} B_{jl} + B_{il} A_{jl}) = 0. \tag{3.24}$$

We determine the vectors in $N(S)$ by evaluating the inner product (3.23) with the rows of $(A|B)$. To get a real code (with an even number of σ_y's), the code should also satisfy

$$\sum_{l=1}^{n} A_{il} B_{il} = 0. \tag{3.25}$$

Another formalism highlights connections with the classical theory of codes over the field GF(4) [26]. This is a field of characteristic two containing four elements, which can be written $\{0, 1, \omega, \omega^2\}$. Since the field has characteristic two,

$$1 + 1 = \omega + \omega = \omega^2 + \omega^2 = 0. \tag{3.26}$$

Also, $\omega^3 = 1$ and $1 + \omega = \omega^2$. We can rewrite the generators as an n-dimensional "vector" over GF(4) by substituting 1 for σ_x, ω for σ_z, and ω^2 for σ_y. The multiplicative structure of $\mathcal{G}$ becomes the additive structure of GF(4). I put vector in quotes because the code need not have the structure of a vector space over GF(4). If it does (that is, the stabilizer is closed under multiplication by

ω), the code is a *linear* code, which is essentially a classical code over GF(4). The most general quantum code is sometimes called an *additive* code, because the stabilizer is only closed under sums of its elements. In this formalism, the five-qubit code appears as

$$\begin{pmatrix} 1 & \omega & \omega & 1 & 0 \\ 0 & 1 & \omega & \omega & 1 \\ 1 & 0 & 1 & \omega & \omega \\ \omega & 1 & 0 & 1 & \omega \end{pmatrix}. \tag{3.27}$$

Note that the five-qubit code is a linear quantum code.

Again, there is an additional condition for a quantum code. Define the "trace" operator by Tr ω = Tr $\omega^2 = 1$, Tr 1 = Tr 0 = 0. Two operators in $\mathcal{G}$ commute iff their images, the vectors u and v over GF(4), satisfy

$$\text{Tr } u \cdot \overline{v} = \text{Tr} \left(\sum_{j=1}^{n} u_j \overline{v}_j \right) = 0, \tag{3.28}$$

where $\overline{v}_j$ is conjugation on the jth component of v, switching ω and ω^2, and leaving 0 and 1 alone.

3.5 Making New Codes From Old Codes

Using old codes to find new ones can simplify the task of finding codes, which can otherwise be quite a difficult problem. There are a number of simple modifications we can make to existing codes to produce new codes with different parameters [25, 26].

One trivial change is to perform a permutation of σ_x, σ_y, and σ_z on each qubit. This leaves the distance and size of the code the same, although it may be useful for codes that can correct different numbers of σ_x, σ_y, and σ_z errors. A slightly less trivial manipulation is to add a new qubit and a new generator which is σ_x for the new qubit. The other generators are tensored with the identity on the new qubit to form the generators of the new code. This makes an $[n, k, d]$ code (degenerate or nondegenerate) into an $[n + 1, k, d]$ degenerate code: Any operator acting as σ_y or σ_z on the new qubit will anticommute with the new generator, and any operator with the form $M \otimes \sigma_{x(n+1)}$ will be equivalent to the operator $M \otimes I$. Therefore, an operator must have at least weight d when restricted to the first n qubits to be in $N(S) - S$.

A less trivial manipulation is to remove the last qubit, converting an $[n, k, d]$ code into an $[n-1, k+1, d-1]$ code. To do this, we choose the $n-k$ generators of S so that M_1 ends σ_x, M_2 ends σ_z, and M_3 through M_{n-k} end I. We can always do this when $d > 1$ by picking the first two and then multiplying by combinations of them to make the others end appropriately.[1] Then the new

[1] If the code has been formed by adding a single σ_x (or σ_y or σ_z) generator, as above, we may not be able to do this for a given qubit, but there will always be at least one qubit for which we can.

M_1'	σ_x	σ_z	σ_z	σ_x
M_2'	σ_y	σ_x	σ_x	σ_y
$\overline{X}_1$	σ_x	σ_x	σ_x	σ_x
$\overline{X}_2$	σ_x	I	σ_x	σ_z
$\overline{Z}_1$	σ_y	σ_z	σ_y	I
$\overline{Z}_2$	I	σ_x	σ_z	σ_z

Table 3.5: A $[4, 2, 2]$ code derived from the $[5, 1, 3]$ code.

code has a stabilizer formed from the last $n - k - 2$ generators, dropping M_1 and M_2. Suppose we have an operator A on the first $n - 1$ qubits of weight w that commutes with M_3 through M_{n-k}. There are four possibilities, all of which lead to an operator of weight at most $w + 1$ that commutes with the original stabilizer:

1. A commutes with both M_1 and M_2.
2. A commutes with M_1, but not M_2. Then $A \otimes \sigma_{xn}$ commutes with M_1 and M_2.
3. A commutes with M_2, but not M_1. Then $A \otimes \sigma_{zn}$ commutes with M_1 and M_2.
4. A anticommutes with both M_1 and M_2. Then $A \otimes \sigma_{yn}$ commutes with M_1 and M_2.

Since the original code had distance d, w must be at least $d-1$, which is therefore the distance of the new code. The stabilizer has $n-k-2$ generators, so the code encodes $(n-1)-(n-k-2) = k+1$ qubits. The new $\overline{X}$ and $\overline{Z}$ operators are M_1 and M_2 (in either order), restricted to the first $n-1$ qubits. An example of this construction is to remove the last qubit from the $[5, 1, 3]$ code of figure 4.2 to produce a $[4, 2, 2]$ code: the generators of the new code are M_1 and M_3M_4, both without the last qubit. The new stabilizer is given in figure 3.5. Note that the $\overline{Z}_1$ operator is equal to $M_3\overline{Z}$ for the five-qubit code. I have multiplied by M_3 so that $\overline{Z}_1$ anticommutes with $\overline{X}_1$.

Another way to make new codes is by *pasting* together old codes. Suppose we have four stabilizers R_1, R_2, S_1, and S_2, with $R_1 \subset S_1$ and $R_2 \subset S_2$. Let R_1 define an $[n_1, l_1, c_1]$ code, R_2 be an $[n_2, l_2, c_2]$ code, S_1 be an $[n_1, k_1, d_1]$ code, and S_2 be an $[n_2, k_2, d_2]$ code. Then $k_i < l_i$ and $c_i \leq d_i$. We require $l_1 - k_1 = l_2 - k_2$ and for S_1 and S_2 to be nondegenerate.[2] Let generators of R_1 be $\{M_1, \ldots, M_{n_1-l_1}\}$, the generators of S_1 be $\{M_1, \ldots, M_{n_1-k_1}\}$, the generators of R_2 be $\{N_1, \ldots, N_{n_2-l_2}\}$, and the generators of S_2 be $\{N_1, \ldots, N_{n_2-k_2}\}$. We form a new stabilizer S on $n_1 + n_2$ qubits generated by

$$\begin{aligned}\{&M_1 \otimes I, \ldots, M_{n_1-l_1} \otimes I, I \otimes N_1, \ldots, I \otimes N_{n_2-l_2},\\ &M_{n_1-l_1+1} \otimes N_{n_2-l_2+1}, \ldots, M_{n_1-k_1} \otimes N_{n_2-k_2}\}.\end{aligned} \tag{3.29}$$

[2] We can actually allow S_1 and S_2 to be degenerate, as long as all the degenerate operators are confined to R_1 and R_2

M_1	σ_x	σ_x	σ_x	σ_x	σ_x	σ_x	σ_x	σ_x	I	I	I	I	I
M_2	σ_z	σ_z	σ_z	σ_z	σ_z	σ_z	σ_z	σ_z	I	I	I	I	I
M_3	I	I	I	I	I	I	I	I	σ_x	σ_z	σ_z	σ_x	I
M_4	I	σ_x	I	σ_x	σ_y	σ_z	σ_y	σ_z	I	σ_x	σ_z	σ_z	σ_x
M_5	I	σ_x	σ_z	σ_y	I	σ_x	σ_z	σ_y	σ_x	I	σ_x	σ_z	σ_z
M_6	I	σ_y	σ_x	σ_z	σ_x	σ_z	I	σ_y	σ_z	σ_x	I	σ_x	σ_z

Table 3.6: The thirteen-qubit code formed by pasting together the five- and eight-qubit codes.

The code has $(n_1 - l_1) + (n_2 - l_2) + (l_i - k_i)$ generators, and therefore encodes $l_1 + k_2 = l_2 + k_1$ qubits. For instance, if S_1 is the eight-qubit code and S_2 is the five-qubit code, with R_1 generated by $\sigma_x\sigma_x\sigma_x\sigma_x\sigma_x\sigma_x\sigma_x\sigma_x$ and $\sigma_z\sigma_z\sigma_z\sigma_z\sigma_z\sigma_z\sigma_z\sigma_z$ and R_2 generated by $\sigma_x\sigma_z\sigma_z\sigma_x I$, we can make the $[13, 7, 3]$ code given in table 3.6.

In general, the distance of the new code will be $\min\{d_1, d_2, c_1 + c_2\}$. This is because an operator acting on just the first n_1 qubits can only commute with S if it commutes with S_1, an operator acting on the last n_2 qubits can only commute with S if it commutes with S_2, and an operator acting on both parts must commute with both $R_1 \otimes I$ and $I \otimes R_2$.

Another very useful way of producing new codes is to *concatenate* two codes to produce a code of greater total distance. Suppose we have an $[n_1, k, d_1]$ code (stabilizer S_1) and we encode each of its n_1 qubits again using an $[n_2, 1, d_2]$ code (stabilizer S_2). The result is an $[n_1 n_2, k, d_1 d_2]$ code. Its stabilizer S is n_1 copies of S_2, acting on the physical qubits in blocks of size n_2, plus an additional $n_1 - k$ generators corresponding to the generators of S_1. However, these generators are encoded to act on the second code. That is, a σ_x acting on the first code must be replaced by an $\overline{X}$ for the second code. For instance, the code resulting from concatenating the five-qubit code with itself has the stabilizer given in table 3.7. The concatenated code has distance $d_1 d_2$ because operators in $N(S) - S$ must have distance at least d_2 on at least d_1 blocks of n_2 qubits, so have weight at least $d_1 d_2$. Note that it is not strictly necessary to use the same code to encode each qubit of S_1.

There are two possible ways to concatenate when S_2 encodes multiple qubits. Suppose S_1 is an $[n_1, k_1, d_1]$ code and S_2 is an $[n_2, k_2, d_2]$ code. Further, suppose n_1 is a multiple of k_2. Then we can encode blocks of S_1 of size k_2 using S_2. This will result in a code using $n_1 n_2 / k_2$ qubits to encode k_1 qubits. It still takes an operator of distance at least d_2 to cause an error on an n_2-qubit block, but such an error can cause up to k_2 errors on S_1, so the resulting code need only have distance $\lceil d_1/k_2 \rceil d_2$. However, the k_2 errors that result are not a general set of k_2 errors, so the code may actually be better. Suppose S_1 has distance d_1' ($d_1' \geq \lceil d_1/k_2 \rceil$) for blocks of k_2 errors, i.e., d_1' such blocks must have errors before the code fails. Then the concatenated code has distance $d_1' d_2$.

Another way to concatenate codes encoding multiple qubits is to add additional blocks of S_1 to fill the spaces in S_2. That is, we actually encode k_2 copies

M_1	σ_x	σ_z	σ_z	σ_x	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
M_2	I	σ_x	σ_z	σ_z	σ_x	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
M_3	σ_x	I	σ_x	σ_z	σ_z	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
M_4	σ_z	σ_x	I	σ_x	σ_z	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
M_5	I	I	I	I	I	σ_x	σ_z	σ_z	σ_x	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
M_6	I	I	I	I	I	I	σ_x	σ_z	σ_z	σ_x	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
M_7	I	I	I	I	I	σ_x	I	σ_x	σ_z	σ_z	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
M_8	I	I	I	I	I	σ_z	σ_x	I	σ_x	σ_z	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
M_9	I	I	I	I	I	I	I	I	I	I	σ_x	σ_z	σ_z	σ_x	I	I	I	I	I	I	I	I	I	I	I
M_{10}	I	I	I	I	I	I	I	I	I	I	I	σ_x	σ_z	σ_z	σ_x	I	I	I	I	I	I	I	I	I	I
M_{11}	I	I	I	I	I	I	I	I	I	I	σ_x	I	σ_x	σ_z	σ_z	I	I	I	I	I	I	I	I	I	I
M_{12}	I	I	I	I	I	I	I	I	I	I	σ_z	σ_x	I	σ_x	σ_z	I	I	I	I	I	I	I	I	I	I
M_{13}	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	σ_x	σ_z	σ_z	σ_x	I	I	I	I	I	I
M_{14}	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	σ_x	σ_z	σ_z	σ_x	I	I	I	I	I
M_{15}	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	σ_x	I	σ_x	σ_z	σ_z	I	I	I	I	I
M_{16}	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	σ_z	σ_x	I	σ_x	σ_z	I	I	I	I	I
M_{17}	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	σ_x	σ_z	σ_z	σ_x	I
M_{18}	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	σ_x	σ_z	σ_z	σ_x
M_{19}	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	σ_x	I	σ_x	σ_z	σ_z
M_{20}	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	σ_z	σ_x	I	σ_x	σ_z
M_{21}	σ_x	σ_x	σ_x	σ_x	σ_x	σ_z	σ_z	σ_z	σ_z	σ_z	σ_z	σ_z	σ_z	σ_z	σ_z	σ_x	σ_x	σ_x	σ_x	σ_x	I	I	I	I	I
M_{22}	I	I	I	I	I	σ_x	σ_x	σ_x	σ_x	σ_x	σ_z	σ_z	σ_z	σ_z	σ_z	σ_z	σ_z	σ_z	σ_z	σ_z	σ_x	σ_x	σ_x	σ_x	σ_x
M_{23}	σ_x	σ_x	σ_x	σ_x	σ_x	I	I	I	I	I	σ_x	σ_x	σ_x	σ_x	σ_x	σ_z	σ_z	σ_z	σ_z	σ_z	σ_z	σ_z	σ_z	σ_z	σ_z
M_{24}	σ_z	σ_z	σ_z	σ_z	σ_z	σ_x	σ_x	σ_x	σ_x	σ_x	I	I	I	I	I	σ_x	σ_x	σ_x	σ_x	σ_x	σ_z	σ_z	σ_z	σ_z	σ_z

Table 3.7: Result of concatenating the five-qubit code with itself.

of S_1, encoding the ith qubit of each copy in the same S_2 block. This produces an $[n_1n_2, k_1k_2, d_1d_2]$ code, since any failure of an S_2 block only produces one error in each S_1 block.

3.6 Higher Dimensional States

So far, we have only considered systems for which the Hilbert space is the tensor product of two-state systems. However, it may turn out that a good physical implementation of quantum computation uses three- or four-level atoms, or spin-one particles, or some other system where it makes more sense to consider it as the tensor product of d-dimensional systems, where $d > 2$. I will call the fundamental unit of such a system a *qudit*. In such a case, we will want to consider error correcting codes where a single qudit error can occur with reasonable probability. For these systems, the stabilizer code formalism needs to be modified to deal with the extra dimensions.

Fundamental to the success of the stabilizer formalism was the use of the Pauli spin matrix basis for possible errors. The algebraic properties of this basis allowed a straightforward characterization of errors depending on whether they commuted or anticommuted with elements of an Abelian group. Knill [31] has codified the properties necessary for this construction to generalize to d-dimensional spaces. Suppose we have a set of d^2 unitary operators $E_1, \ldots, E_{n^2}$ (including the identity) acting on a single qudit such that the E_i's form a basis for all possible $d \times d$ complex matrices. If $E_iE_j = w_{ij}E_{i*j}$ for all i, j (where $*$ is some binary group operation), then the E_i's are said to form a *nice* error basis. The values w_{ij} will then have modulus one. Given a nice error basis, we form the group $\mathcal{G}_n$ for this basis as the tensor product of n copies of the error basis, with possible overall phases generated by the w_{ij}'s. Then an Abelian subgroup S of $\mathcal{G}_n$ that does not contain any nontrivial phase times the identity will have a nontrivial set T of states in the Hilbert space in the $+1$ eigenspace of every operator in S. The code T can detect any error E for which $EM = cME$ for some $M \in \mathcal{G}_n$ and some $c \neq 1$.

One interesting complication of codes over d-dimensional spaces is that when S has $n - k$ generators, T need not encode k qudits. This can only occur when d is composite and the order of a generator of S is a nontrivial factor of d. It is still true that if S has r elements, then T will be (d^n/r)-dimensional. If all the generators of S have order d, T does encode k qudits.

One particularly convenient error basis for any d is generated by D_ω and C_n, where $(D_\omega)_{ij} = \delta_{ij}\omega^i$ and $(C_n)_{ij} = \delta_{j,(i+1 \bmod n)}$. ω is a primitive nth root of unity. For $d = 2$, this just reduces to the usual Pauli basis, since $C_2 = \sigma_x$ and $D_{-1} = \sigma_z$. For higher d, D_ω maps $|i\rangle \to \omega^i|i\rangle$ and C_n adds one modulo n. This is a nice error basis, with

$$C_nD_\omega = \omega D_\omega C_n. \tag{3.30}$$

The elements of the basis can be written $C_n^a D_\omega^b$, and

$$\left(C_n^a D_\omega^b\right)\left(C_n^c D_\omega^d\right) = \omega^{ad-bc}\left(C_n^c D_\omega^d\right)\left(C_n^a D_\omega^b\right). \tag{3.31}$$

Codes for higher-dimensional systems have not been as extensively studied as those for two-dimensional systems, but some constructions are given in [31, 32, 33, 34, 35].